Package 'fastai'

November 12, 2020

Type Package

Title Interface to 'fastai'

Version 2.0.1

Maintainer Turgut Abdullayev <turqut.a.314@gmail.com>

Description The 'fastai' <https://docs.fast.ai/index.html> library simplifies training fast and accurate neural networks using modern best practices. It is based on research in to deep learning best practices undertaken at 'fast.ai', including 'out of the box' support for vision, text, tabular, audio, time series, and collaborative filtering models.

License Apache License 2.0

URL https://github.com/henry090/fastai

BugReports https://github.com/henry090/fastai/issues

Encoding UTF-8

LazyData true

RoxygenNote 7.1.1

Imports reticulate, generics, png, ggplot2, ggpubr, glue

Suggests knitr, testthat, rmarkdown, curl, magrittr, data.table, vctrs, stats, utils, R.utils

VignetteBuilder knitr

NeedsCompilation no

Author Turgut Abdullayev [ctb, cre, cph, aut]

Repository CRAN

Date/Publication 2020-11-12 08:50:03 UTC

R topics documented:

`*.fastai.torch_core.TensorMask` *Multiply*

Description

Multiply

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a * b
```

Arguments

a	tensor
b	tensor

Value

tensor

`+.fastai.torch_core.TensorMask`
Add

Description

Add

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a + b
```

Arguments

a	tensor
b	tensor

Value

tensor

`+.torch.nn.modules.container.Sequential`
Add layers to Sequential

Description

Add layers to Sequential

Usage

```
## S3 method for class 'torch.nn.modules.container.Sequential'
a + b
```

Arguments

a	sequential model
b	layer

Value

model

`/.fastai.torch_core.TensorMask`
Div

Description

Div

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a / b
```

Arguments

a	tensor
b	tensor

Value

tensor

`<.fastai.torch_core.TensorMask`
Less

Description

Less

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a < b
```

Arguments

a	tensor
b	tensor

Value

tensor

`<=.fastai.torch_core.TensorMask`
Less or equal

Description

Less or equal

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a <= b
```

Arguments

a	tensor
b	tensor

Value

tensor

`==.fastai.torch_core.TensorMask`
Equal

Description

Equal

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a == b
```

Arguments

a	tensor
b	tensor

Value

tensor

Examples

```
aa = tensor(1:10)
aa == aa
```

`==.torch.Tensor` *Equal*

Description

Equal

Usage

```
## S3 method for class 'torch.Tensor'
a == b
```

Arguments

a	tensor
b	tensor

Value

tensor

`>.fastai.torch_core.TensorMask` *Greater*

Description

Greater

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a > b
```

Arguments

a	tensor
b	tensor

Value

tensor

`>=.fastai.torch_core.TensorMask` *Greater or equal*

Description

Greater or equal

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a >= b
```

Arguments

a	tensor
b	tensor

Value

tensor

Examples

```
aa = tensor(1:10)
aa >= aa
```

`abs` *Abs*

Description

Abs

Usage

```
## S3 method for class 'torch.Tensor'
abs(x)
```

Arguments

x	tensor

Value

tensor

`abs.fastai.torch_core.TensorMask`
Abs

Description

Abs

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
abs(x)
```

Arguments

x	tensor

Value

tensor

Examples

```
aa = tensor(-1:-10)
abs(aa)
```

AccumMetric *AccumMetric*

Description

Stores predictions and targets on CPU in accumulate to perform final calculations with ‘func‘.

Usage

```
AccumMetric(
  func,
  dim_argmax = NULL,
  activation = "no",
  thresh = NULL,
  to_np = FALSE,
  invert_arg = FALSE,
  flatten = TRUE,
  ...
)
```

Arguments

func	function
dim_argmax	dimension argmax
activation	activation
thresh	threshold point
to_np	to matrix or not
invert_arg	invert arguments
flatten	flatten
...	additional arguments to pass

Value

None

accuracy *Accuracy*

Description

Compute accuracy with ‘targ‘ when ‘pred‘ is bs * n_classes

Usage

```
accuracy(inp, targ, axis = -1)
```

Arguments

inp	predictions
targ	targets
axis	axis

Value

None

accuracy_multi *Accuracy_multi*

Description

Compute accuracy when ‘inp‘ and ‘targ‘ are the same size.

Usage

```
accuracy_multi(inp, targ, thresh = 0.5, sigmoid = TRUE)
```

Arguments

inp	predictions
targ	targets
thresh	threshold point
sigmoid	sigmoid

Value

None

accuracy_thresh_expand *Accuracy threshold expand*

Description

Compute accuracy after expanding ‘y_true‘ to the size of ‘y_pred‘.

Usage

```
accuracy_thresh_expand(y_pred, y_true, thresh = 0.5, sigmoid = TRUE)
```

Arguments

y_pred	predictions
y_true	actuals
thresh	threshold point
sigmoid	sigmoid function

Value

None

Adam *Adam*

Description

Adam

Usage

```
Adam(...)
```

Arguments

...	parameters to pass

Value

None

adam_step *Adam_step*

Description

Step for Adam with ‘lr‘ on ‘p‘

Usage

```
adam_step(p, lr, mom, step, sqr_mom, grad_avg, sqr_avg, eps, ...)
```

Arguments

p	p
lr	learning rate
mom	momentum
step	step
sqr_mom	sqr momentum
grad_avg	grad average
sqr_avg	sqr average
eps	epsilon
...	additional arguments to pass

Value

None

AdaptiveAvgPool *AdaptiveAvgPool*

Description

nn$AdaptiveAvgPool layer for ‘ndim‘

Usage

```
AdaptiveAvgPool(sz = 1, ndim = 2)
```

Arguments

sz	size
ndim	dimension size

AdaptiveConcatPool1d *AdaptiveConcatPool1d*

Description

Layer that concats 'AdaptiveAvgPool1d' and 'AdaptiveMaxPool1d'

Usage

```
AdaptiveConcatPool1d(size = NULL)
```

Arguments

size	output size

Value

None

AdaptiveConcatPool2d *AdaptiveConcatPool2d*

Description

Layer that concats 'AdaptiveAvgPool2d' and 'AdaptiveMaxPool2d'

Usage

```
AdaptiveConcatPool2d(size = NULL)
```

Arguments

size	output size

Value

None

AdaptiveGANSwitcher *Adaptive GAN Switcher*

Description

Switcher that goes back to generator/critic when the loss goes below ‘gen_thresh‘/‘crit_thresh‘.

Usage

```
AdaptiveGANSwitcher(gen_thresh = NULL, critic_thresh = NULL)
```

Arguments

gen_thresh	generator threshold
critic_thresh	discriminator threshold

Value

None

AdaptiveLoss *AdaptiveLoss*

Description

Expand the ‘target‘ to match the ‘output‘ size before applying ‘crit‘.

Usage

```
AdaptiveLoss(crit)
```

Arguments

crit	critic

Value

Loss object

adaptive_pool *Adaptive_pool*

Description

Adaptive_pool

Usage

```
adaptive_pool(pool_type)
```

Arguments

pool_type	pooling type

Value

Nonee

add *Add*

Description

Add

Sinh

Usage

```
## S3 method for class 'torch.Tensor'
a + b

## S3 method for class 'torch.Tensor'
sinh(x)
```

Arguments

a	tensor
b	tensor
x	tensor

Value

tensor

tensor

AddChannels *Add Channels*

Description

Add ‘n_dim‘ channels at the end of the input.

Usage

```
AddChannels(n_dim)
```

Arguments

n_dim	number of dimensions

AddNoise *Add Noise*

Description

Adds noise of specified color and level to the audio signal

Usage

```
AddNoise(noise_level = 0.05, color = 0)
```

Arguments

noise_level	noise level
color	int, color

Value

None

add_cyclic_datepart *Add cyclic datepart*

Description

Helper function that adds trigonometric date/time features to a date in the column ‘field_name‘ of ‘df‘.

Usage

```
add_cyclic_datepart(
  df,
  field_name,
  prefix = NULL,
  drop = TRUE,
  time = FALSE,
  add_linear = FALSE
)
```

Arguments

df	df
field_name	field_name
prefix	prefix
drop	drop
time	time
add_linear	add_linear

Value

data frame

add_datepart *Add datepart*

Description

Helper function that adds columns relevant to a date in the column ‘field_name‘ of ‘df‘.

Usage

```
add_datepart(df, field_name, prefix = NULL, drop = TRUE, time = FALSE)
```

Arguments

df	df
field_name	field_name
prefix	prefix
drop	drop
time	time

Value

data frame

`AffineCoordTfm` *AffineCoordTfm*

Description

Combine and apply affine and coord transforms

Usage

```
AffineCoordTfm(
  aff_fs = NULL,
  coord_fs = NULL,
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  mode_mask = "nearest",
  align_corners = NULL
)
```

Arguments

aff_fs	aff fs
coord_fs	coordinate fs
size	size
mode	mode
pad_mode	padding mode
mode_mask	mode mask
align_corners	align corners

Value

None

affine_coord *Aaffine_coord*

Description

Aaffine_coord

Usage

```
affine_coord(
  x,
  mat = NULL,
  coord_tfm = NULL,
  sz = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  align_corners = TRUE,
  ...
)
```

Arguments

x	tensor
mat	mat
coord_tfm	coordinate tfm
sz	sz
mode	mode
pad_mode	padding mode
align_corners	align corners
...	additional arguments

Value

None

affine_mat *Affline mat*

Description

Affline mat

Usage

```
affine_mat(...)
```

Arguments

...	parameters to pass

Value

None

alexnet *Alexnet*

Description

AlexNet model architecture

Usage

```
alexnet(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"One weird trick..." <https://arxiv.org/abs/1404.5997>

Value

model

Examples

```
## Not run:

alexnet(pretrained = FALSE, progress = TRUE)


## End(Not run)
```

`apply_perspective` *Apply_perspective*

Description

Apply perspective tranfom on ‘coords‘ with ‘coeffs‘

Usage

```
apply_perspective(coords, coeffs)
```

Arguments

`coords`	coordinates
`coeffs`	coefficient

Value

None

`APScoreBinary` *APScoreBinary*

Description

Average Precision for single-label binary classification problems

Usage

```
APScoreBinary(
  axis = -1,
  average = "macro",
  pos_label = 1,
  sample_weight = NULL
)
```

Arguments

axis	axis
average	average
pos_label	pos_label
sample_weight	sample_weight

Value

None

`APScoreMulti` *APScoreMulti*

Description

Average Precision for multi-label classification problems

Usage

```
APScoreMulti(
  sigmoid = TRUE,
  average = "macro",
  pos_label = 1,
  sample_weight = NULL
)
```

Arguments

sigmoid	sigmoid
average	average
pos_label	pos_label
sample_weight	sample_weight

Value

None

aspect *Aspect*

Description

Aspect

Usage

```
aspect(img)
```

Arguments

img	image

Value

None

AudioBlock *AudioBlock*

Description

A 'TransformBlock' for audios

Usage

```
AudioBlock(
  cache_folder = NULL,
  sample_rate = 16000,
  force_mono = TRUE,
  crop_signal_to = NULL
)
```

Arguments

cache_folder	cache folder
sample_rate	sample rate
force_mono	force mono or not
crop_signal_to	int, crop signal

Value

None

AudioBlock_from_folder *AudioBlock from folder*

Description

Build a ‘AudioBlock‘ from a ‘path‘ and caches some intermediary results

Usage

```
AudioBlock_from_folder(
  path,
  sample_rate = 16000,
  force_mono = TRUE,
  crop_signal_to = NULL
)
```

Arguments

path	directory, path
sample_rate	sample rate
force_mono	force mono or not
crop_signal_to	int, crop signal

Value

None

AudioGetter *AudioGetter*

Description

Create ‘get_audio_files‘ partial function that searches path suffix ‘suf‘

Usage

```
AudioGetter(suf = "", recurse = TRUE, folders = NULL)
```

Arguments

suf	suffix
recurse	recursive or not
folders	vector, folders

Details

and passes along ‘kwargs‘, only in ‘folders‘, if specified.

Value

None

`AudioPadType` *AudioPadType module*

Description

AudioPadType module

Usage

```
AudioPadType()
```

Value

None

`AudioSpectrogram` *AudioSpectrogram module*

Description

AudioSpectrogram module

Usage

```
AudioSpectrogram()
```

Value

None

AudioTensor *Audio Tensor*

Description

Semantic torch tensor that represents an audio.

Usage

```
AudioTensor(x, sr = NULL)
```

Arguments

x	tensor
sr	sr

Value

tensor

AudioTensor_create *AudioTensor create*

Description

Creates audio tensor from file

Usage

```
AudioTensor_create(
  fn,
  cache_folder = NULL,
  out = NULL,
  normalization = TRUE,
  channels_first = TRUE,
  num_frames = 0,
  offset = 0,
  signalinfo = NULL,
  encodinginfo = NULL,
  filetype = NULL
)
```

Arguments

fn	function
cache_folder	cache folder
out	out format
normalization	apply normalization or not
channels_first	channels first/last
num_frames	number of frames
offset	offset
signalinfo	signal info
encodinginfo	encoding info
filetype	the type of file

Value

None

AudioToMFCC *AudioToMFCC*

Description

Transform to create MFCC features from audio tensors.

Usage

```
AudioToMFCC(
  sample_rate = 16000,
  n_mfcc = 40,
  dct_type = 2,
  norm = "ortho",
  log_mels = FALSE,
  melkwargs = NULL
)
```

Arguments

sample_rate	sample rate
n_mfcc	number of mel-frequency cepstral coefficients
dct_type	dct type
norm	normalization type
log_mels	apply log to mels
melkwargs	additional arguments for mels

Value

None

AudioToMFCC_from_cfg *AudioToMFCC from cfg*

Description

Creates AudioToMFCC from configuration file

Usage

```
AudioToMFCC_from_cfg(audio_cfg)
```

Arguments

audio_cfg	audio configuration

Value

None

AudioToSpec_from_cfg *AudioToSpec from cfg*

Description

Creates AudioToSpec from configuration file

Usage

```
AudioToSpec_from_cfg(audio_cfg)
```

Arguments

audio_cfg	audio configuration

Value

None

audio_extensions *Audio_extensions*

Description

get all allowed audio extensions

Usage

```
audio_extensions()
```

Value

vector

aug_transforms *Augmentation*

Description

Utility func to easily create a list of flip, rotate, zoom, warp, lighting transforms.

Usage

```
aug_transforms(
  mult = 1,
  do_flip = TRUE,
  flip_vert = FALSE,
  max_rotate = 10,
  min_zoom = 1,
  max_zoom = 1.1,
  max_lighting = 0.2,
  max_warp = 0.2,
  p_affine = 0.75,
  p_lighting = 0.75,
  xtra_tfms = NULL,
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  align_corners = TRUE,
  batch = FALSE,
  min_scale = 1
)
```

Arguments

mult	ratio
do_flip	to do flip
flip_vert	flip vertical or not
max_rotate	maximum rotation
min_zoom	minimum zoom
max_zoom	maximum zoom
max_lighting	maximum lighting
max_warp	maximum warp
p_affine	probability affine
p_lighting	probability lighting
xtra_tfms	extra transformations
size	size of image
mode	mode
pad_mode	padding mode
align_corners	align_corners
batch	batch size
min_scale	minimum scale

Value

None

Examples

```
## Not run:

URLs_PETS()

path = 'oxford-iiit-pet'

path_img = 'oxford-iiit-pet/images'
fnames = get_image_files(path_img)

dls = ImageDataLoaders_from_name_re(
path, fnames, pat='(.+)_.jpg$',
item_tfms=Resize(size = 460), bs = 10,
batch_tfms=list(aug_transforms(size = 224, min_scale = 0.75),
                Normalize_from_stats( imagenet_stats() )
)
)


## End(Not run)
```

average_grad *Average_grad*

Description

Keeps track of the avg grads of ‘p‘ in ‘state‘ with ‘mom‘.

Usage

```
average_grad(p, mom, dampening = FALSE, grad_avg = NULL, ...)
```

Arguments

p	p
mom	momentum
dampening	dampening
grad_avg	grad average
...	additional args to pass

Value

None

average_sqr_grad *Average_sqr_grad*

Description

Average_sqr_grad

Usage

```
average_sqr_grad(p, sqr_mom, dampening = TRUE, sqr_avg = NULL, ...)
```

Arguments

p	p
sqr_mom	sqr momentum
dampening	dampening
sqr_avg	sqr average
...	additional args to pass

Value

None

AvgLoss *AvgLoss*

Description

Flattens input and output, same as nn$AvgLoss

Usage

```
AvgLoss(...)
```

Arguments

...	parameters to pass

Value

Loss object

AvgPool *AvgPool*

Description

nn$AvgPool layer for ‘ndim‘

Usage

```
AvgPool(ks = 2, stride = NULL, padding = 0, ndim = 2, ceil_mode = FALSE)
```

Arguments

ks	kernel size
stride	the stride of the window. Default value is kernel_size
padding	implicit zero padding to be added on both sides
ndim	dimension number
ceil_mode	when True, will use ceil instead of floor to compute the output shape

Value

None

AvgSmoothLoss *AvgSmoothLoss*

Description

Smooth average of the losses (exponentially weighted with ‘beta‘)

Usage

```
AvgSmoothLoss(beta = 0.98)
```

Arguments

beta	beta, defaults to 0.98

Value

Loss object

AWD_LSTM *AWD_LSTM*

Description

AWD-LSTM inspired by https://arxiv.org/abs/1708.02182

Usage

```
AWD_LSTM(
  vocab_sz,
  emb_sz,
  n_hid,
  n_layers,
  pad_token = 1,
  hidden_p = 0.2,
  input_p = 0.6,
  embed_p = 0.1,
  weight_p = 0.5,
  bidir = FALSE
)
```

Arguments

vocab_sz	vocab_sz
emb_sz	emb_sz
n_hid	n_hid
n_layers	n_layers
pad_token	pad_token
hidden_p	hidden_p
input_p	input_p
embed_p	embed_p
weight_p	weight_p
bidir	bidir

Value

None

---

awd_lstm_clas_split *Awd_lstm_clas_split*

Description

Split a RNN ‘model‘ in groups for differential learning rates.

Usage

```
awd_lstm_clas_split(model)
```

Arguments

model	model

Value

None

awd_lstm_lm_split *Awd_lstm_lm_split*

Description

Split a RNN ‘model‘ in groups for differential learning rates.

Usage

```
awd_lstm_lm_split(model)
```

Arguments

model	model

Value

None

AWD_QRNN *AWD_QRNN*

Description

Same as an AWD-LSTM, but using QRNNs instead of LSTMs

Usage

```
AWD_QRNN(
  vocab_sz,
  emb_sz,
  n_hid,
  n_layers,
  pad_token = 1,
  hidden_p = 0.2,
  input_p = 0.6,
  embed_p = 0.1,
  weight_p = 0.5,
  bidir = FALSE
)
```

Arguments

vocab_sz	vocab_sz
emb_sz	emb_sz
n_hid	n_hid
n_layers	n_layers
pad_token	pad_token
hidden_p	hidden_p
input_p	input_p
embed_p	embed_p
weight_p	weight_p
bidir	bidir

Value

None

BalancedAccuracy *BalancedAccuracy*

Description

Balanced Accuracy for single-label binary classification problems

Usage

```
BalancedAccuracy(axis = -1, sample_weight = NULL, adjusted = FALSE)
```

Arguments

axis	axis
sample_weight	sample_weight
adjusted	adjusted

References

None

BaseLoss *BaseLoss*

Description

Flattens input and output, same as nn$BaseLoss

Usage

```
BaseLoss(...)
```

Arguments

...	parameters to pass

Value

Loss object

BaseTokenizer *BaseTokenizer*

Description

Basic tokenizer that just splits on spaces

Usage

```
BaseTokenizer(split_char = " ")
```

Arguments

split_char	separator

Value

None

`BasicMelSpectrogram` *BasicMelSpectrogram*

Description

BasicMelSpectrogram

Usage

```
BasicMelSpectrogram(
  sample_rate = 16000,
  n_fft = 400,
  win_length = NULL,
  hop_length = NULL,
  f_min = 0,
  f_max = NULL,
  pad = 0,
  n_mels = 128,
  window_fn = torch()$hann_window,
  power = 2,
  normalized = FALSE,
  wkwargs = NULL,
  mel = TRUE,
  to_db = TRUE
)
```

Arguments

`sample_rate`	sample rate
`n_fft`	number of fast fourier transforms
`win_length`	windowing length
`hop_length`	hopping length
`f_min`	minimum frequency
`f_max`	maximum frequency
`pad`	padding
`n_mels`	number of mel-spectrograms
`window_fn`	window function
`power`	power
`normalized`	normalized or not
`wkwargs`	additional arguments
`mel`	mel-spectrogram or not
`to_db`	to decibels

Value

None

`BasicMFCC` *Basic MFCC*

Description

Basic MFCC

Usage

```
BasicMFCC(
  sample_rate = 16000,
  n_mfcc = 40,
  dct_type = 2,
  norm = "ortho",
  log_mels = FALSE,
  melkwargs = NULL
)
```

Arguments

`sample_rate`	sample rate
`n_mfcc`	number of mel-frequency cepstral coefficients
`dct_type`	dct type
`norm`	normalization type
`log_mels`	apply log to mels
`melkwargs`	additional arguments for mels

Value

None

BasicSpectrogram *BasicSpectrogram*

Description

BasicSpectrogram

Usage

```
BasicSpectrogram(
  n_fft = 400,
  win_length = NULL,
  hop_length = NULL,
  pad = 0,
  window_fn = torch()$hann_window,
  power = 2,
  normalized = FALSE,
  wkwargs = NULL,
  mel = FALSE,
  to_db = TRUE
)
```

Arguments

n_fft	number of fast fourier transforms
win_length	windowing length
hop_length	hopping length
pad	padding mode
window_fn	window function
power	power
normalized	normalized or not
wkwargs	additional arguments
mel	mel-spectrogram or not
to_db	to decibels

Value

None

basic_critic *Basic critic*

Description

A basic critic for images ‘n_channels‘ x ‘in_size‘ x ‘in_size‘.

Usage

```
basic_critic(in_size, n_channels, ...)
```

Arguments

in_size	input size
n_channels	The number of channels
...	additional parameters to pass

Value

None

Examples

```
## Not run:

critic    = basic_critic(in_size = 64, n_channels = 3, n_extra_layers = 1,
                         act_cls = partial(nn$LeakyReLU, negative_slope = 0.2))


## End(Not run)
```

basic_generator *Basic generator*

Description

A basic generator from ‘in_sz‘ to images ‘n_channels‘ x ‘out_size‘ x ‘out_size‘.

Usage

```
basic_generator(out_size, n_channels, ...)
```

Arguments

out_size	out_size
n_channels	n_channels
...	additional params to pass

Value

generator object

Examples

```
## Not run:

generator = basic_generator(out_size = 64, n_channels = 3, n_extra_layers = 1)


## End(Not run)
```

BatchNorm *BatchNorm*

Description

BatchNorm layer with ‘nf‘ features and ‘ndim‘ initialized depending on ‘norm_type‘.

Usage

```
BatchNorm(
  nf,
  ndim = 2,
  norm_type = 1,
  eps = 1e-05,
  momentum = 0.1,
  affine = TRUE,
  track_running_stats = TRUE
)
```

Arguments

nf	input shape
ndim	dimension number
norm_type	normalization type
eps	epsilon
momentum	momentum
affine	affine
track_running_stats	track running statistics

Value

None

`BatchNorm1dFlat` *BatchNorm1dFlat*

Description

'nn.BatchNorm1d', but first flattens leading dimensions

Usage

```
BatchNorm1dFlat(
  num_features,
  eps = 1e-05,
  momentum = 0.1,
  affine = TRUE,
  track_running_stats = TRUE
)
```

Arguments

`num_features`	number of features
`eps`	epsilon
`momentum`	momentum
`affine`	affine
`track_running_stats`	track running statistics

Value

None

`BBoxBlock` *BBoxBlock*

Description

A 'TransformBlock' for bounding boxes in an image

Usage

```
BBoxBlock()
```

Value

None

BBoxLabeler *BBoxLabeler*

Description

Delegates (‘__call__‘,‘decode‘,‘setup‘) to (‘encodes‘,‘decodes‘,‘setups‘) if ‘split_idx‘ matches

Usage

```
BBoxLabeler(enc = NULL, dec = NULL, split_idx = NULL, order = NULL)
```

Arguments

enc	encoder
dec	decoder
split_idx	split by index
order	order

Value

None

BBoxLblBlock *BBoxLblBlock*

Description

A ‘TransformBlock‘ for labeled bounding boxes, potentially with ‘vocab‘

Usage

```
BBoxLblBlock(vocab = NULL, add_na = TRUE)
```

Arguments

vocab	vocabulary
add_na	add NA

Value

None’

Examples

```
## Not run:

URLs_COCO_TINY()

c(images, lbl_bbox) %<-% get_annotations('coco_tiny/train.json')
timg = Transform(ImageBW_create)
idx = 49
c(coco_fn,bbox) %<-% list(paste('coco_tiny/train',images[[idx]],sep = '/'),
                          lbl_bbox[[idx]])
coco_img = timg(coco_fn)

tbbox = LabeledBBox(TensorBBox(bbox[[1]]), bbox[[2]])

coco_bb = function(x) {
TensorBBox_create(bbox[[1]])
}

coco_lbl = function(x) {
  bbox[[2]]
}

coco_dsrc = Datasets(c(rep(coco_fn,10)),
                     list(Image_create(), list(coco_bb),
                          list( coco_lbl, MultiCategorize(add_na = TRUE) )
                     ), n_inp = 1)

coco_tdl = TfmdDL(coco_dsrc, bs = 9,
                  after_item = list(BBoxLabeler(), PointScaler(),
                                    ToTensor()),
                  after_batch = list(IntToFloatTensor(), aug_transforms())
)

coco_tdl %>% show_batch(dpi = 200)


## End(Not run)
```

---

bb_pad *Bb_pad*

Description

Function that collect ‘samples‘ of labelled bboxes and adds padding with ‘pad_idx‘.

Usage

```
bb_pad(samples, pad_idx = 0)
```

Arguments

samples	samples
pad_idx	pad index

Value

None

BCELossFlat *BCELossFlat*

Description

Flattens input and output, same as nn$BCELoss

Usage

```
BCELossFlat(...)
```

Arguments

...	parameters to pass

Value

Loss object

BCEWithLogitsLossFlat *BCEWithLogitsLossFlat*

Description

BCEWithLogitsLossFlat

Usage

```
BCEWithLogitsLossFlat(...)
```

Arguments

...	parameters to pass

Value

Loss object

blurr *Blurr module*

Description

Blurr module

Usage

```
blurr()
```

Value

None

BrierScore *BrierScore*

Description

Brier score for single-label classification problems

Usage

```
BrierScore(axis = -1, sample_weight = NULL, pos_label = NULL)
```

Arguments

axis	axis
sample_weight	sample_weight
pos_label	pos_label

Value

None

BrierScoreMulti *BrierScoreMulti*

Description

Brier score for multi-label classification problems

Usage

```
BrierScoreMulti(
  thresh = 0.5,
  sigmoid = TRUE,
  sample_weight = NULL,
  pos_label = NULL
)
```

Arguments

thresh	thresh
sigmoid	sigmoid
sample_weight	sample_weight
pos_label	pos_label

Value

None

bs_find *Bs_find*

Description

Launch a mock training to find a good batch size to minimize training time.

Usage

```
bs_find(
  object,
  lr,
  num_it = NULL,
  n_batch = 5,
  simulate_multi_gpus = TRUE,
  show_plot = TRUE
)
```

Arguments

object	model/learner
lr	learning rate
num_it	number of iterations
n_batch	number of batches
simulate_multi_gpus	simulate on multi gpus or not
show_plot	show plot or not

Details

However, it may not be a good batch size to minimize the validation loss. A good batch size is where the Simple Noise Scale converge ignoring the small growing trend with the number of iterations if exists. The optimal batch size is about an order the magnitud where Simple Noise scale converge. Typically, the optimal batch size in image classification problems will be 2-3 times lower where

---

bs_finder *Bs finder*

Description

Bs finder

Usage

```
bs_finder()
```

Value

None

---

bt *Builtins module*

Description

Builtins module

Usage

```
bt()
```

Value

None

Callback *Callback module*

Description

Callback module

Usage

```
Callback()
```

Value

None

Cat *Cat*

Description

Concatenate layers outputs over a given dim

Usage

```
Cat(layers, dim = 1)
```

Arguments

layers	layers
dim	dimension size

Value

None

catalyst *Catalyst module*

Description

Catalyst module

Usage

```
catalyst()
```

Value

None

catalyst_model *Catalyst model*

Description

Catalyst model

Usage

```
catalyst_model()
```

Value

model

Categorify *Categorify*

Description

Transform the categorical variables to that type.

Usage

```
Categorify(cat_names, cont_names)
```

Arguments

cat_names	The names of the categorical variables
cont_names	The names of the continuous variables

Value

None

`CategoryBlock` *CategoryBlock*

Description

'TransformBlock' for single-label categorical targets

Usage

```
CategoryBlock(vocab = NULL, sort = TRUE, add_na = FALSE)
```

Arguments

vocab	vocabulary
sort	sort or not
add_na	add NA

Value

Block object

`ceiling.fastai.torch_core.TensorMask`
Ceil

Description

Ceil

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
ceiling(x)
```

Arguments

x	tensor

Value

tensor

ceiling_ *Ceil*

Description

Ceil

Usage

```
## S3 method for class 'torch.Tensor'
ceiling(x)
```

Arguments

x	tensor

Value

tensor

ChangeVolume *Change Volume*

Description

Changes the volume of the signal

Usage

```
ChangeVolume(p = 0.5, lower = 0.5, upper = 1.5)
```

Arguments

p	probability
lower	lower bound
upper	upper bound

Value

None

`children_and_parameters` *Children_and_parameters*

Description

Return the children of ‘m‘ and its direct parameters not registered in modules.

Usage

```
children_and_parameters(m)
```

Arguments

`m`	parameters

Value

None

`ClassificationInterpretation_from_learner` *ClassificationInterpretation_from_learner*

Description

Construct interpretation object from a learner

Usage

```
ClassificationInterpretation_from_learner(
  learn,
  ds_idx = 1,
  dl = NULL,
  act = NULL
)
```

Arguments

`learn`	learner/model
`ds_idx`	ds by index
`dl`	DL application
`act`	activation

Value

interpretation object

clean_raw_keys *Clean_raw_keys*

Description

Clean_raw_keys

Usage

```
clean_raw_keys(wgts)
```

Arguments

wgts	wgts

Value

None

clip_remove_empty *Clip_remove_empty*

Description

Clip bounding boxes with image border and label background the empty ones

Usage

```
clip_remove_empty(bbox, label)
```

Arguments

bbox	bbox
label	label

Value

None

cm *Cm module*

Description

Cm module

Usage

```
cm()
```

Value

None

cnn_config *Cnn config*

Description

Convenience function to easily create a config for ‘create_cnn_model‘

Usage

```
cnn_config(
  cut = NULL,
  pretrained = TRUE,
  n_in = 3,
  init = nn$init$kaiming_normal_,
  custom_head = NULL,
  concat_pool = TRUE,
  lin_ftrs = NULL,
  ps = 0.5,
  bn_final = FALSE,
  lin_first = FALSE,
  y_range = NULL
)
```

Arguments

cut	cut
pretrained	pre-trained or not
n_in	input shape
init	initializer

custom_head	custom head
concat_pool	concatenate pooling
lin_ftrs	linear filters
ps	parameter server
bn_final	batch normalization final
lin_first	linear first
y_range	y_range

Value

None

---

cnn_learner *Cnn_learner*

Description

Build a convnet style learner from ‘dls‘ and ‘arch‘

Usage

```
cnn_learner(
  dls,
  arch,
  loss_func = NULL,
  pretrained = TRUE,
  cut = NULL,
  splitter = NULL,
  y_range = NULL,
  config = NULL,
  n_out = NULL,
  normalize = TRUE,
  opt_func = Adam(),
  lr = 0.001,
  cbs = NULL,
  metrics = NULL,
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE,
  moms = list(0.95, 0.85, 0.95)
)
```

Arguments

dls	data loader object
arch	a model architecture
loss_func	loss function
pretrained	pre-trained or not
cut	cut
splitter	It is a function that takes self.model and returns a list of parameter groups (or just one parameter group if there are no different parameter groups).
y_range	y_range
config	configuration
n_out	the number of out
normalize	normalize
opt_func	The function used to create the optimizer
lr	learning rate
cbs	Cbs is one or a list of Callbacks to pass to the Learner.
metrics	It is an optional list of metrics, that can be either functions or Metrics.
path	The folder where to work
model_dir	Path and model_dir are used to save and/or load models.
wd	It is the default weight decay used when training the model.
wd_bn_bias	It controls if weight decay is applied to BatchNorm layers and bias.
train_bn	It controls if BatchNorm layers are trained even when they are supposed to be frozen according to the splitter.
moms	The default momentums used in Learner.fit_one_cycle.

Value

learner object

Examples

```
## Not run:

URLs_MNIST_SAMPLE()
# transformations
tfms = aug_transforms(do_flip = FALSE)
path = 'mnist_sample'
bs = 20

#load into memory
data = ImageDataLoaders_from_folder(path, batch_tfms = tfms, size = 26, bs = bs)


learn = cnn_learner(data, resnet18(), metrics = accuracy, path = getwd())
```

```
## End(Not run)
```

CohenKappa *CohenKappa*

Description

Cohen kappa for single-label classification problems

Usage

```
CohenKappa(axis = -1, labels = NULL, weights = NULL, sample_weight = NULL)
```

Arguments

axis	axis
labels	labels
weights	weights
sample_weight	sample_weight

Value

None

---

collab *Collab module*

Description

Collab module

Usage

```
collab()
```

Value

None

`CollabDataLoaders_from_dblock` *CollabDataLoaders_from_dblock*

Description

Create a dataloaders from a given ‘dblock‘

Usage

```
CollabDataLoaders_from_dblock(
  dblock,
  source,
  path = ".",
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL
)
```

Arguments

`dblock`	dblock
`source`	source
`path`	The folder where to work
`bs`	The batch size
`val_bs`	The batch size for the validation DataLoader (defaults to bs)
`shuffle_train`	If we shuffle the training DataLoader or not
`device`	device

Value

None

`CollabDataLoaders_from_df` *CollabDataLoaders_from_df*

Description

Create a ‘DataLoaders‘ suitable for collaborative filtering from ‘ratings‘.

Usage

```
CollabDataLoaders_from_df(
  ratings,
  valid_pct = 0.2,
  user_name = NULL,
  item_name = NULL,
  rating_name = NULL,
  seed = NULL,
  path = ".",
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL
)
```

Arguments

ratings	ratings
valid_pct	The random percentage of the dataset to set aside for validation (with an optional seed)
user_name	The name of the column containing the user (defaults to the first column)
item_name	The name of the column containing the item (defaults to the second column)
rating_name	The name of the column containing the rating (defaults to the third column)
seed	random seed
path	The folder where to work
bs	The batch size
val_bs	The batch size for the validation DataLoader (defaults to bs)
shuffle_train	If we shuffle the training DataLoader or not
device	the device, e.g. cpu, cuda, and etc.

Value

None

Examples

```
## Not run:

URLs_MOVIE_LENS_ML_100k()
c(user,item,title)  %<-% list('userId','movieId','title')
ratings = fread('ml-100k/u.data', col.names = c(user,item,'rating','timestamp'))
movies = fread('ml-100k/u.item', col.names = c(item, 'title', 'date', 'N', 'url',
                                               paste('g',1:19,sep = '')))
rating_movie = ratings[movies[, .SD, .SDcols=c(item,title)], on = item]
dls = CollabDataLoaders_from_df(rating_movie, seed = 42, valid_pct = 0.1, bs = 64,
```

```
item_name=title, path='ml-100k')


## End(Not run)
```

`collab_learner` *Collab_learner*

Description

Create a Learner for collaborative filtering on 'dls'.

Usage

```
collab_learner(
  dls,
  n_factors = 50,
  use_nn = FALSE,
  emb_szs = NULL,
  layers = NULL,
  config = NULL,
  y_range = NULL,
  loss_func = NULL,
  opt_func = Adam(),
  lr = 0.001,
  splitter = trainable_params(),
  cbs = NULL,
  metrics = NULL,
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE,
  moms = list(0.95, 0.85, 0.95)
)
```

Arguments

`dls`	a data loader object
`n_factors`	The number of factors
`use_nn`	use_nn
`emb_szs`	embedding size
`layers`	list of layers
`config`	configuration

y_range	y_range
loss_func	It can be any loss function you like. It needs to be one of fastai's if you want to use Learn.predict or Learn.get_preds, or you will have to implement special methods (see more details after the BaseLoss documentation).
opt_func	The function used to create the optimizer
lr	learning rate
splitter	It is a function that takes self.model and returns a list of parameter groups (or just one parameter group if there are no different parameter groups).
cbs	Cbs is one or a list of Callbacks to pass to the Learner.
metrics	It is an optional list of metrics, that can be either functions or Metrics.
path	The folder where to work
model_dir	Path and model_dir are used to save and/or load models.
wd	It is the default weight decay used when training the model.
wd_bn_bias	It controls if weight decay is applied to BatchNorm layers and bias.
train_bn	It controls if BatchNorm layers are trained even when they are supposed to be frozen according to the splitter.
moms	The default momentums used in Learner.fit_one_cycle.

Value

learner object

Examples

```
## Not run:

URLs_MOVIE_LENS_ML_100k()
c(user,item,title)  %<-% list('userId','movieId','title')
ratings = fread('ml-100k/u.data', col.names = c(user,item,'rating','timestamp'))
movies = fread('ml-100k/u.item', col.names = c(item, 'title', 'date', 'N', 'url',
                                               paste('g',1:19,sep = '')))
rating_movie = ratings[movies[, .SD, .SDcols=c(item,title)], on = item]
dls = CollabDataLoaders_from_df(rating_movie, seed = 42, valid_pct = 0.1, bs = 64,
item_name=title, path='ml-100k')

learn = collab_learner(dls, n_factors = 40, y_range=c(0, 5.5))

learn %>% fit_one_cycle(1, 5e-3,  wd = 1e-1)


## End(Not run)
```

CollectDataCallback *CollectDataCallback*

Description

CollectDataCallback

Usage

```
CollectDataCallback(...)
```

Arguments

...	parameters to pass

Value

None

colors *Colors module*

Description

Colors module

Usage

```
colors()
```

Value

None

ColReader *ColReader*

Description

Read ‘cols‘ in ‘row‘ with potential ‘pref‘ and ‘suff‘

Usage

```
ColReader(cols, pref = "", suff = "", label_delim = NULL)
```

Arguments

cols	columns
pref	pref
suff	suffix
label_delim	label separator

Value

None

ColSplitter *ColSplitter*

Description

Split ‘items‘ (supposed to be a dataframe) by value in ‘col‘

Usage

```
ColSplitter(col = "is_valid")
```

Arguments

col	column

Value

None

combined_flat_anneal *Combined_flat_anneal*

Description

Create a schedule with constant learning rate ‘start_lr‘ for ‘pct‘ proportion of the training, and a ‘curve_type‘ learning rate (till ‘end_lr‘) for remaining portion of training.

Usage

```
combined_flat_anneal(pct, start_lr, end_lr = 0, curve_type = "linear")
```

Arguments

pct	Proportion of training with a constant learning rate.
start_lr	Desired starting learning rate, used for beginnning pct of training.
end_lr	Desired end learning rate, training will conclude at this learning rate.
curve_type	Curve type for learning rate annealing. Options are ’linear’, ’cosine’, and ’exponential’.

competitions_list *Competitions list*

Description

Competitions list

Usage

```
competitions_list(
  group = NULL,
  category = NULL,
  sort_by = NULL,
  page = 1,
  search = NULL
)
```

Arguments

group	group to filter result to
category	category to filter result to
sort_by	how to sort the result, see valid_competition_sort_by for options
page	the page to return (default is 1)
search	a search term to use (default is empty string)

Value

list of competitions

`competition_download_file` *Competition download file*

Description

download a competition file to a designated location, or use

Usage

```
competition_download_file(
  competition,
  file_name,
  path = NULL,
  force = FALSE,
  quiet = FALSE
)
```

Arguments

`competition`	the name of the competition
`file_name`	the configuration file name
`path`	a path to download the file to
`force`	force the download if the file already exists (default FALSE)
`quiet`	suppress verbose output (default is FALSE)

Value

None

Examples

```
## Not run:

com_nm = 'titanic'

titanic_files = competition_list_files(com_nm)
titanic_files = lapply(1:length(titanic_files),
                       function(x) as.character(titanic_files[[x]]))

str(titanic_files)

if(!dir.exists(com_nm)) {
```

```
 dir.create(com_nm)
}

# download via api
competition_download_files(competition = com_nm, path = com_nm, unzip = TRUE)


## End(Not run)
```

`competition_download_files` *Competition download files*

Description

Competition download files

Usage

```
competition_download_files(
  competition,
  path = NULL,
  force = FALSE,
  quiet = FALSE,
  unzip = FALSE
)
```

Arguments

`competition`	the name of the competition
`path`	a path to download the file to
`force`	force the download if the file already exists (default FALSE)
`quiet`	suppress verbose output (default is TRUE)
`unzip`	unzip downloaded files

Value

None

`competition_leaderboard_download` *Competition leaderboard download*

Description

Download competition leaderboards

Usage

```
competition_leaderboard_download(competition, path, quiet = TRUE)
```

Arguments

`competition`	the name of the competition
`path`	a path to download the file to
`quiet`	suppress verbose output (default is TRUE)

Value

data frame

`competition_list_files` *Competition list files*

Description

list files for competition

Usage

```
competition_list_files(competition)
```

Arguments

`competition`	the name of the competition

Value

list of files

Examples

```
## Not run:

com_nm = 'titanic'
titanic_files = competition_list_files(com_nm)



## End(Not run)
```

`competition_submit` *Competition submit*

Description

Competition submit

Usage

```
competition_submit(file_name, message, competition, quiet = FALSE)
```

Arguments

`file_name`	the competition metadata file
`message`	the submission description
`competition`	the competition name
`quiet`	suppress verbose output (default is FALSE)

Value

None

`Contrast` *Contrast*

Description

Apply change in contrast of ‘max_lighting‘ to batch of images with probability ‘p‘.

Usage

```
Contrast(max_lighting = 0.2, p = 0.75, draw = NULL, batch = FALSE)
```

Arguments

max_lighting	maximum lighting
p	probability
draw	draw
batch	batch

Value

None

ConvLayer *ConvLayer*

Description

Create a sequence of convolutional (‘ni‘ to ‘nf‘), ReLU (if ‘use_activ‘) and ‘norm_type‘ layers.

Usage

```
ConvLayer(
  ni,
  nf,
  ks = 3,
  stride = 1,
  padding = NULL,
  bias = NULL,
  ndim = 2,
  norm_type = 1,
  bn_1st = TRUE,
  act_cls = nn$ReLU,
  transpose = FALSE,
  init = "auto",
  xtra = NULL,
  bias_std = 0.01,
  dilation = 1,
  groups = 1,
  padding_mode = "zeros"
)
```

Arguments

ni	number of inputs
nf	outputs/ number of features
ks	kernel size
stride	stride

padding	padding
bias	bias
ndim	dimension number
norm_type	normalization type
bn_1st	batch normalization 1st
act_cls	activation
transpose	transpose
init	initializer
xtra	xtra
bias_std	bias standard deviation
dilation	specify the dilation rate to use for dilated convolution
groups	groups size
padding_mode	padding mode, e.g 'zeros'

Value

None

---

convT_norm_relu *ConvT_norm_relu*

Description

ConvT_norm_relu

Usage

```
convT_norm_relu(ch_in, ch_out, norm_layer, ks = 3, stride = 2, bias = TRUE)
```

Arguments

ch_in	input
ch_out	output
norm_layer	normalziation layer
ks	kernel size
stride	stride size
bias	bias true or not

Value

None

conv_norm_lr *Conv_norm_lr*

Description

Conv_norm_lr

Usage

```
conv_norm_lr(
  ch_in,
  ch_out,
  norm_layer = NULL,
  ks = 3,
  bias = TRUE,
  pad = 1,
  stride = 1,
  activ = TRUE,
  slope = 0.2,
  init = nn$init$normal_,
  init_gain = 0.02
)
```

Arguments

ch_in	input
ch_out	output
norm_layer	normalziation layer
ks	kernel size
bias	bias
pad	pad
stride	stride
activ	activation
slope	slope
init	inititializer
init_gain	initializer gain

Value

None

CorpusBLEUMetric *CorpusBLEUMetric*

Description

Blueprint for defining a metric

Usage

```
CorpusBLEUMetric(vocab_sz = 5000, axis = -1)
```

Arguments

vocab_sz	vocab_sz
axis	axis

Value

None

---

cos.fastai.torch_core.TensorMask *Cos*

Description

Cos

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
cos(x)
```

Arguments

x	tensor

Value

tensor

cosh.fastai.torch_core.TensorMask *Cosh*

Description

Cosh

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
cosh(x)
```

Arguments

x	tensor

Value

tensor

cosh_ *Cosh*

Description

Cosh

Usage

```
## S3 method for class 'torch.Tensor'
cosh(x)
```

Arguments

x	tensor

Value

tensor

cos_ *Cos*

Description

Cos

Usage

```
## S3 method for class 'torch.Tensor'
cos(x)
```

Arguments

x	tensor

Value

tensor

crap *Crappify module*

Description

Crappify module

Usage

```
crap()
```

Value

None

crappifier *Crappifier*

Description

Crappifier

Usage

```
crappifier(path_lr, path_hr)
```

Arguments

path_lr	path from (origin)
path_hr	path to (destination)

Value

None

Examples

```
## Not run:

items = get_image_files(path_hr)
parallel(crappifier(path_lr, path_hr), items)


## End(Not run)
```

create_body *Create_body*

Description

Cut off the body of a typically pretrained ‘arch‘ as determined by ‘cut‘

Usage

```
create_body(...)
```

Arguments

...	parameters to pass

Value

None

Examples

```
## Not run:

encoder = create_body(resnet34(), pretrained = TRUE)


## End(Not run)
```

`create_cnn_model` *Create_cnn_model*

Description

Create custom convnet architecture using ‘arch‘, ‘n_in‘ and ‘n_out‘

Usage

```
create_cnn_model(
  arch,
  n_out,
  cut = NULL,
  pretrained = TRUE,
  n_in = 3,
  init = nn$init$kaiming_normal_,
  custom_head = NULL,
  concat_pool = TRUE,
  lin_ftrs = NULL,
  ps = 0.5,
  bn_final = FALSE,
  lin_first = FALSE,
  y_range = NULL
)
```

Arguments

`arch`	a model architecture
`n_out`	number of outs
`cut`	cut
`pretrained`	pretrained model or not
`n_in`	input shape

init	initializer
custom_head	custom head
concat_pool	concatenate pooling
lin_ftrs	linear fiters
ps	parameter server
bn_final	batch normalization final
lin_first	linear first
y_range	y_range

Value

None

`create_fcn` *Create_fcn*

Description

A bunch of convolutions stacked together.

Usage

```
create_fcn(ni, nout, ks = 9, conv_sizes = c(128, 256, 128), stride = 1)
```

Arguments

ni	number of input channels
nout	output shape
ks	kernel size
conv_sizes	convolution sizes
stride	stride

Value

model

create_head *Create_head*

Description

Model head that takes ‘nf‘ features, runs through ‘lin_ftrs‘, and out ‘n_out‘ classes.

Usage

```
create_head(
  nf,
  n_out,
  lin_ftrs = NULL,
  ps = 0.5,
  concat_pool = TRUE,
  bn_final = FALSE,
  lin_first = FALSE,
  y_range = NULL
)
```

Arguments

nf	number of features
n_out	number of out features
lin_ftrs	linear features
ps	parameter server
concat_pool	concatenate pooling
bn_final	batch normalization final
lin_first	linear first
y_range	y_range

Value

None

`create_inception` *Create_inception*

Description

Creates an InceptionTime arch from ‘ni‘ channels to ‘nout‘ outputs.

Usage

```
create_inception(
  ni,
  nout,
  kss = c(39, 19, 9),
  depth = 6,
  bottleneck_size = 32,
  nb_filters = 32,
  head = TRUE
)
```

Arguments

`ni`	number of input channels
`nout`	number of outputs, should be equal to the number of classes for classification tasks.
`kss`	kernel sizes for the inception Block.
`depth`	depth
`bottleneck_size`	The number of channels on the convolution bottleneck.
`nb_filters`	Channels on the convolution of each kernel.
`head`	TRUE if we want a head attached.

Value

model

`create_mlp` *Create_mlp*

Description

A simple model builder to create a bunch of BatchNorm1d, Dropout and Linear layers, with “‘act_fn“‘ activations.

Usage

```
create_mlp(ni, nout, linear_sizes = c(500, 500, 500))
```

Arguments

ni	number of input channels
nout	output shape
linear_sizes	linear output sizes

Value

model

---

create_resnet *Create_resnet*

Description

Basic 11 Layer - 1D resnet builder

Usage

```
create_resnet(
  ni,
  nout,
  kss = c(9, 5, 3),
  conv_sizes = c(64, 128, 128),
  stride = 1
)
```

Arguments

ni	number of input channels
nout	output shape
kss	kernel size
conv_sizes	convolution sizes
stride	stride

Value

model

CropPad *CropPad*

Description

Center crop or pad an image to ‘size‘

Usage

```
CropPad(size, pad_mode = "zeros", ...)
```

Arguments

size	size
pad_mode	padding mode
...	additional arguments

Value

None

CropTime *Crop Time*

Description

Random crops full spectrogram to be length specified in ms by crop_duration

Usage

```
CropTime(duration, pad_mode = AudioPadType$Zeros)
```

Arguments

duration	int, duration
pad_mode	padding mode, by default ‘AudioPadType$Zeros‘

Value

None

CrossEntropyLossFlat *CrossEntropyLossFlat*

Description

Same as ‘nn$Module‘, but no need for subclasses to call ‘super().__init__‘

Usage

```
CrossEntropyLossFlat(...)
```

Arguments

...	parameters to pass

Value

Loss object

CSVLogger *CSVLogger*

Description

Basic class handling tweaks of the training loop by changing a ‘Learner‘ in various events

Usage

```
CSVLogger(fname = "history.csv", append = FALSE)
```

Arguments

fname	file name
append	append or not

Value

None

Examples

```
## Not run:

URLs_MNIST_SAMPLE()
# transformations
tfms = aug_transforms(do_flip = FALSE)
path = 'mnist_sample'
bs = 20

#load into memory
data = ImageDataLoaders_from_folder(path, batch_tfms = tfms, size = 26, bs = bs)


learn = cnn_learner(data, resnet18(), metrics = accuracy, path = getwd())

learn %>% fit_one_cycle(2, cbs = CSVLogger())


## End(Not run)
```

`CudaCallback` *CudaCallback*

Description

Move data to CUDA device

Usage

```
CudaCallback(device = NULL)
```

Arguments

`device`	device name

Value

None

cutout_gaussian *Cutout_gaussian*

Description

Replace all ‘areas‘ in ‘x‘ with N(0,1) noise

Usage

```
cutout_gaussian(x, areas)
```

Arguments

x	tensor
areas	areas

Value

None

CycleGAN *CycleGAN*

Description

CycleGAN model.

Usage

```
CycleGAN(
  ch_in = 3,
  ch_out = 3,
  n_features = 64,
  disc_layers = 3,
  gen_blocks = 9,
  lsgan = TRUE,
  drop = 0,
  norm_layer = NULL
)
```

Arguments

ch_in	input
ch_out	output
n_features	number of features
disc_layers	discriminator layers
gen_blocks	generator blocks
lsgan	ls gan
drop	dropout rate
norm_layer	normalziation layer

Details

When called, takes in input batch of real images from both domains and outputs fake images for the opposite domains (with the generators). Also outputs identity images after passing the images into generators that outputs its domain type (needed for identity loss). Attributes: ‘G_A‘ (‘nn.Module‘): takes real input B and generates fake input A ‘G_B‘ (‘nn.Module‘): takes real input A and generates fake input B ‘D_A‘ (‘nn.Module‘): trained to make the difference between real input A and fake input A ‘D_B‘ (‘nn.Module‘): trained to make the difference between real input B and fake input B

Value

None

CycleGANLoss *CycleGANLoss*

Description

CycleGAN loss function. The individual loss terms are also atrributes of this class that are accessed by fastai for recording during training.

Usage

```
CycleGANLoss(cgan, l_A = 10, l_B = 10, l_idt = 0.5, lsgan = TRUE)
```

Arguments

cgan	The CycleGAN model.
l_A	lambda_A, weight of domain A losses. (default=10)
l_B	lambda_B, weight of domain B losses. (default=10)
l_idt	lambda_idt, weight of identity lossees. (default=0.5)
lsgan	Whether or not to use LSGAN objective (default=True)

Details

Attributes: ‘self.cgan‘ (‘nn.Module‘): The CycleGAN model. ‘self.l_A‘ (‘float‘): lambda_A, weight of domain A losses. ‘self.l_B‘ (‘float‘): lambda_B, weight of domain B losses. ‘self.l_idt‘ (‘float‘): lambda_idt, weight of identity lossees. ‘self.crit‘ (‘AdaptiveLoss‘): The adversarial loss function (either a BCE or MSE loss depending on ‘lsgan‘ argument) ‘self.real_A‘ and ‘self.real_B‘ (‘fastai.torch_core.TensorImage‘): Real images from domain A and B. ‘self.id_loss_A‘ (‘torch.FloatTensor‘): The identity loss for domain A calculated in the forward function ‘self.id_loss_B‘ (‘torch.FloatTensor‘): The identity loss for domain B calculated in the forward function ‘self.gen_loss‘ (‘torch.FloatTensor‘): The generator loss calculated in the forward function ‘self.cyc_loss‘ (‘torch.FloatTensor‘): The cyclic loss calculated in the forward function

---

CycleGANTrainer *CycleGANTrainer*

Description

Learner Callback for training a CycleGAN model.

Usage

```
CycleGANTrainer(...)
```

Arguments

...	parameters to pass

Value

None

---

cycle_learner *Cycle_learner*

Description

Initialize and return a ‘Learner‘ object with the data in ‘dls‘, CycleGAN model ‘m‘, optimizer function ‘opt_func‘, metrics ‘metrics‘,

Usage

```
cycle_learner(
  dls,
  m,
  opt_func = Adam(),
  show_imgs = TRUE,
  imgA = TRUE,
  imgB = TRUE,
  show_img_interval = 10,
  ...
)
```

Arguments

dls	dataloader
m	CycleGAN model
opt_func	optimizer
show_imgs	show images
imgA	image a (from)
imgB	image B (to)
show_img_interval	show images interval rafe
...	additional arguments

Details

and callbacks ‘cbs‘. Additionally, if ‘show_imgs‘ is TRUE, it will show intermediate predictions during training. It will show domain B-to-A predictions if ‘imgA‘ is TRUE and/or domain A-to-B predictions if ‘imgB‘ is TRUE. Additionally, it will show images every ‘show_img_interval‘ epochs. ‘ Other ‘Learner‘ arguments can be passed as well.

Value

None

DataBlock *DataBlock*

Description

Generic container to quickly build ‘Datasets‘ and ‘DataLoaders‘

Usage

```
DataBlock(
  blocks = NULL,
  dl_type = NULL,
  getters = NULL,
  n_inp = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  ...
)
```

Arguments

blocks	input blocks
dl_type	DL application
getters	how to get dataet
n_inp	n_inp is the number of elements in the tuples that should be considered part of the input and will default to 1 if tfms consists of one set of transforms
item_tfms	One or several transforms applied to the items before batching them
batch_tfms	One or several transforms applied to the batches once they are formed
...	additional parameters to pass

Value

Block object

---

dataloaders *Dataloaders from dls object*

Description

Create a ‘DataLoaders‘ object from ‘source‘

Usage

```
dataloaders(object, ...)
```

Arguments

object	model
...	additional parameters to pass

Examples

```
## Not run:

dls = TabularDataTable(df, procs, cat_names, cont_names,
y_names = dep_var, splits = list(tr_idx, ts_idx) ) %>%
  dataloaders(bs = 50)


## End(Not run)
```

`Datasets` *Datasets*

Description

A dataset that creates a list from each ‘tfms‘, passed thru ‘item_tfms‘

Usage

```
Datasets(
  items = NULL,
  tfms = NULL,
  tls = NULL,
  n_inp = NULL,
  dl_type = NULL,
  use_list = NULL,
  do_setup = TRUE,
  split_idx = NULL,
  train_setup = TRUE,
  splits = NULL,
  types = NULL,
  verbose = FALSE
)
```

Arguments

`items`	items
`tfms`	transformations
`tls`	tls
`n_inp`	n_inp
`dl_type`	DL type
`use_list`	use list
`do_setup`	do setup
`split_idx`	split by index

train_setup	train setup
splits	splits
types	types
verbose	verbose

Value

None

`Data_Loaders` *Data Loaders*

Description

Data Loaders

Usage

```
Data_Loaders(...)
```

Arguments

...	parameters to pass

Value

loader object

Examples

```
## Not run:

data = Data_Loaders(train_loader, test_loader)

learn = Learner(data, Net(), loss_func = F$nll_loss,
                opt_func = Adam(), metrics = accuracy, cbs = CudaCallback())

learn %>% fit_one_cycle(1, 1e-2)


## End(Not run)
```

dcmread *Read dicom*

Description

Open a 'DICOM' file

Usage

```
dcmread(fn, force = FALSE)
```

Arguments

fn	file name
force	logical, force

Value

dicom object

Examples

```
## Not run:

img = dcmread('hemorrhage.dcm')


## End(Not run)
```

debias *Debias*

Description

Debias

Usage

```
debias(mom, damp, step)
```

Arguments

mom	mom
damp	damp
step	step

Value

None

`Debugger` *Debugger*

Description

A module to debug inside a model

Usage

```
Debugger(...)
```

Arguments

`...`	parameters to pass

Value

None

`decision_plot` *Decision_plot*

Description

Visualizes a model's decisions using cumulative SHAP values.

Usage

```
decision_plot(object, class_id = 0, row_idx = -1, dpi = 200, ...)
```

Arguments

`object`	ShapInterpretation object
`class_id`	is used to indicate the class of interest for a classification model. It can either be an int or str representation for a class of choice. Each colored line in the plot represents the model's prediction for a single observation.
`row_idx`	If no index is passed in to use from the data, it will default to the first ten samples on the test set. Note:plotting too many samples at once can make the plot illegible.
`dpi`	dots per inch
`...`	additional arguments

Value

None

decode_spec_tokens *Decode_spec_tokens*

Description

Decode the special tokens in ‘tokens‘

Usage

```
decode_spec_tokens(tokens)
```

Arguments

tokens	tokens

Value

None

default_split *Default_split*

Description

Default split of a model between body and head

Usage

```
default_split(m)
```

Arguments

m	parameters

Value

None

Delta *Delta*

Description

Creates delta with order 1 and 2 from spectrogram and concatenate with the original

Usage

```
Delta(width = 9)
```

Arguments

width	int, width

Value

None

densenet121 *Densenet121*

Description

Densenet121

Usage

```
densenet121(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Densely Connected Convolutional Networks" <https://arxiv.org/pdf/1608.06993.pdf>

Value

model

densenet161 *Densenet161*

Description

Densenet161

Usage

```
densenet161(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Densely Connected Convolutional Networks" <https://arxiv.org/pdf/1608.06993.pdf>

Value

model

densenet169 *Densenet169*

Description

Densenet169

Usage

```
densenet169(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Densely Connected Convolutional Networks" <https://arxiv.org/pdf/1608.06993.pdf>

Value

model

densenet201 *Densenet201*

Description

Densenet201

Usage

```
densenet201(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Densely Connected Convolutional Networks" <https://arxiv.org/pdf/1608.06993.pdf>

Value

model

DenseResBlock *Dense Res Block*

Description

Resnet block of ‘nf‘ features. ‘conv_kwargs‘ are passed to ‘conv_layer‘.

Usage

```
DenseResBlock(
  nf,
  norm_type = 1,
  ks = 3,
  stride = 1,
  padding = NULL,
  bias = NULL,
  ndim = 2,
  bn_1st = TRUE,
  act_cls = nn$ReLU,
  transpose = FALSE,
  init = "auto",
  xtra = NULL,
```

```
  bias_std = 0.01,
  dilation = 1,
  groups = 1,
  padding_mode = "zeros"
)
```

Arguments

nf	number of features
norm_type	normalization type
ks	kernel size
stride	stride
padding	padding
bias	bias
ndim	number of dimensions
bn_1st	batch normalization 1st
act_cls	activation
transpose	transpose
init	initizalier
xtra	xtra
bias_std	bias standard deviation
dilation	dilation number
groups	groups number
padding_mode	padding mode

Value

block

---

dependence_plot *Dependence_plot*

Description

Plots the value of a variable on the x-axis and the SHAP value of the same variable on the y-axis. Accepts a class_id and variable_name.

Usage

```
dependence_plot(object, variable_name = "", class_id = 0, dpi = 200, ...)
```

Arguments

object	ShapInterpretation object
variable_name	the name of the column
class_id	is used to indicate the class of interest for a classification model. It can either be an int or str representation for a class of choice. This plot shows how the model depends on the given variable. Vertical dispersion of the datapoints represent interaction effects. Gray ticks along the y-axis are datapoints where the variable's values were NaN.
dpi	dots per inch
...	additional arguments

Value

None

DeterministicDihedral *DeterministicDihedral*

Description

Apply a random dihedral transformation to a batch of images with a probability ‘p‘

Usage

```
DeterministicDihedral(
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  align_corners = NULL
)
```

Arguments

size	size
mode	mode
pad_mode	padding mode
align_corners	align corners

Value

None

DeterministicDraw *DeterministicDraw*

Description

DeterministicDraw

Usage

```
DeterministicDraw(vals)
```

Arguments

vals	values

Value

None

DeterministicFlip *DeterministicFlip*

Description

Flip the batch every other call

Usage

```
DeterministicFlip(
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  align_corners = TRUE,
  ...
)
```

Arguments

size	size
mode	mode
pad_mode	padding mode
align_corners	align corners
...	parameters to pass

Value

None

detuplify_pg *Detuplify_pg*

Description

Detuplify_pg

Usage

```
detuplify_pg(d)
```

Arguments

d	d

Value

None

Dice *Dice coefficient*

Description

Dice coefficient metric for binary target in segmentation

Usage

```
Dice(axis = 1)
```

Arguments

axis	axis

Value

None

Dicom *Dicom class*

Description

Dicom class

Usage

```
Dicom()
```

Value

None

dicom_windows *Dicom_windows module*

Description

Dicom_windows module

Usage

```
dicom_windows()
```

Value

None

Dihedral *Dihedral*

Description

Apply a random dihedral transformation to a batch of images with a probability ‘p‘

Apply a random dihedral transformation to a batch of images with a probability ‘p‘

Usage

```
Dihedral(
  p = 0.5,
  draw = NULL,
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  align_corners = NULL,
  batch = FALSE
)

Dihedral(
  p = 0.5,
  draw = NULL,
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  align_corners = NULL,
  batch = FALSE
)
```

Arguments

p	probability
draw	draw
size	size
mode	mode
pad_mode	padding mode
align_corners	align corners
batch	batch

Value

None

None

`DihedralItem` *DihedralItem*

Description

Randomly flip with probability ‘p‘

Usage

```
DihedralItem(p = 1, nm = NULL, before_call = NULL)
```

Arguments

p	probability
nm	nm
before_call	before call

Value

None

---

dihedral_mat *Dihedral_mat*

Description

Return a random dihedral matrix

Usage

```
dihedral_mat(x, p = 0.5, draw = NULL, batch = FALSE)
```

Arguments

x	tensor
p	probability
draw	draw
batch	batch

Value

None

---

dim *Dim*

Description

Dim

Usage

```
## S3 method for class 'torch.Tensor'
dim(x)
```

Arguments

x	tensor

Value

tensor

`dim.fastai.torch_core.TensorMask` *Dim*

Description

Dim

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
dim(x)
```

Arguments

x	tensor

Value

tensor

`discriminator` *Discriminator*

Description

Discriminator

Usage

```
discriminator(
  ch_in,
  n_ftrs = 64,
  n_layers = 3,
  norm_layer = NULL,
  sigmoid = FALSE
)
```

Arguments

ch_in	input
n_ftrs	number of filters
n_layers	number of layers
norm_layer	normalization layer
sigmoid	apply sigmoid function or not

---

div *Div*

Description

Div

Usage

```
## S3 method for class 'torch.Tensor'
a / b
```

Arguments

a	tensor
b	tensor

Value

tensor

DownmixMono *Downmix Mono*

Description

Transform multichannel audios into single channel

Usage

```
DownmixMono(enc = NULL, dec = NULL, split_idx = NULL, order = NULL)
```

Arguments

enc	encoder
dec	decoder
split_idx	split by index
order	order, by default is NULL

Value

None

`dropout_mask` *Dropout_mask*

Description

Return a dropout mask of the same type as ‘x‘, size ‘sz‘, with probability ‘p‘ to cancel an element.

Usage

```
dropout_mask(x, sz, p)
```

Arguments

x	x
sz	sz
p	p

Value

None

`DynamicUnet` *DynamicUnet*

Description

Create a U-Net from a given architecture.

Usage

```
DynamicUnet(
  encoder,
  n_classes,
  img_size,
  blur = FALSE,
  blur_final = TRUE,
  self_attention = FALSE,
  y_range = NULL,
  last_cross = TRUE,
  bottle = FALSE,
  act_cls = nn()$ReLU,
  init = nn()$init$kaiming_normal_,
  norm_type = NULL
)
```

Arguments

encoder	encoder
n_classes	number of classes
img_size	image size
blur	blur is used to avoid checkerboard artifacts at each layer.
blur_final	blur final is specific to the last layer.
self_attention	self_attention determines if we use a self attention layer at the third block before the end.
y_range	If y_range is passed, the last activations go through a sigmoid rescaled to that range.
last_cross	last cross
bottle	bottle
act_cls	activation
init	initializer
norm_type	normalization type

Value

None

EarlyStoppingCallback *EarlyStoppingCallback*

Description

EarlyStoppingCallback

Usage

```
EarlyStoppingCallback(...)
```

Arguments

...	parameters to pass

Value

None

`Embedding` *Embedding*

Description

Embedding layer with truncated normal initialization

Usage

```
Embedding(ni, nf)
```

Arguments

`ni`	inputs
`nf`	outputs / number of features

Value

None

`EmbeddingDropout` *EmbeddingDropout*

Description

Apply dropout with probability ‘embed_p‘ to an embedding layer ‘emb‘.

Usage

```
EmbeddingDropout(emb, embed_p)
```

Arguments

`emb`	emb
`embed_p`	embed_p

Value

None

emb_sz_rule *Emb_sz_rule*

Description

Rule of thumb to pick embedding size corresponding to ‘n_cat‘

Usage

```
emb_sz_rule(n_cat)
```

Arguments

n_cat	n_cat

Value

None

error_rate *Error rate*

Description

1 - ‘accuracy‘

Usage

```
error_rate(inp, targ, axis = -1)
```

Arguments

inp	The predictions of the model
targ	The corresponding labels
axis	Axis

Value

tensor

Examples

```
## Not run:

learn = cnn_learner(dls, resnet34(), metrics = error_rate)


## End(Not run)
```

---

exp *Exp*

Description

Exp

Usage

```
## S3 method for class 'torch.Tensor'
exp(x)
```

Arguments

x	tensor

Value

tensor

---

exp.fastai.torch_core.TensorMask
Exp

Description

Exp

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
exp(x)
```

Arguments

x	tensor

Value

tensor

ExplainedVariance *Explained Variance*

Description

Explained variance between predictions and targets

Usage

```
ExplainedVariance(sample_weight = NULL)
```

Arguments

sample_weight	sample_weight

Value

None

---

expm1 *Expm1*

Description

Expm1

Usage

```
## S3 method for class 'torch.Tensor'
expm1(x)
```

Arguments

x	tensor

Value

tensor

expm1.fastai.torch_core.TensorMask *Expm1*

Description

Expm1

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
expm1(x)
```

Arguments

x	tensor

Value

tensor

export_generator *Export_generator*

Description

Export_generator

Usage

```
export_generator(
  learn,
  generator_name = "generator",
  path = ".",
  convert_to = "B"
)
```

Arguments

learn	learner/model
generator_name	generator name
path	path (save dir)
convert_to	convert to

Value

None

exp_rmspe *Exp_rmspe*

Description

Root mean square percentage error of the exponential of predictions and targets

Usage

```
exp_rmspe(preds, targs)
```

Arguments

preds	predicitons
targs	targets

Value

None

F1Score *F1Score*

Description

F1 score for single-label classification problems

Usage

```
F1Score(
  axis = -1,
  labels = NULL,
  pos_label = 1,
  average = "binary",
  sample_weight = NULL
)
```

Arguments

axis	axis
labels	labels
pos_label	pos_label
average	average
sample_weight	sample_weight

Value

None

`F1ScoreMulti` *F1ScoreMulti*

Description

F1 score for multi-label classification problems

Usage

```
F1ScoreMulti(
  thresh = 0.5,
  sigmoid = TRUE,
  labels = NULL,
  pos_label = 1,
  average = "macro",
  sample_weight = NULL
)
```

Arguments

`thresh`	thresh
`sigmoid`	sigmoid
`labels`	labels
`pos_label`	pos_label
`average`	average
`sample_weight`	sample_weight

Value

None

fastaudio *Fastaudio module*

Description

Fastaudio module

Usage

```
fastaudio()
```

Value

None

fastinf *Wandb module*

Description

Wandb module

Usage

```
fastinf()
```

Value

None

fa_collate *Fa_collate*

Description

Fa_collate

Usage

```
fa_collate(t)
```

Arguments

t	text

Value

None

fa_convert *Da_convert*

Description

Da_convert

Usage

```
fa_convert(t)
```

Arguments

t	text

Value

None

FBeta *FBeta*

Description

FBeta score with ‘beta‘ for single-label classification problems

Usage

```
FBeta(
  beta,
  axis = -1,
  labels = NULL,
  pos_label = 1,
  average = "binary",
  sample_weight = NULL
)
```

Arguments

beta	beta
axis	axis
labels	labels
pos_label	pos_label
average	average
sample_weight	sample_weight

Value

None

`FBetaMulti` *FBetaMulti*

Description

FBeta score with ‘beta‘ for multi-label classification problems

Usage

```
FBetaMulti(
  beta,
  thresh = 0.5,
  sigmoid = TRUE,
  labels = NULL,
  pos_label = 1,
  average = "macro",
  sample_weight = NULL
)
```

Arguments

`beta`	beta
`thresh`	thresh
`sigmoid`	sigmoid
`labels`	labels
`pos_label`	pos_label
`average`	average
`sample_weight`	sample_weight

Value

None

FetchPredsCallback *FetchPredsCallback*

Description

A callback to fetch predictions during the training loop

Usage

```
FetchPredsCallback(
  ds_idx = 1,
  dl = NULL,
  with_input = FALSE,
  with_decoded = FALSE,
  cbs = NULL,
  reorder = TRUE
)
```

Arguments

ds_idx	dataset index
dl	DL application
with_input	with input or not
with_decoded	with decoded or not
cbs	callbacks
reorder	reorder or not

Value

None

FileSplitter *File Splitter*

Description

Split ‘items‘ by providing file ‘fname‘ (contains names of valid items separated by newline).

Usage

```
FileSplitter(fname)
```

Arguments

fname	file name

Value

None

`FillMissing` *Fill Missing*

Description

Fill the missing values in continuous columns.

Usage

```
FillMissing(
  cat_names,
  cont_names,
  fill_strategy = FillStrategy_MEDIAN(),
  add_col = TRUE,
  fill_val = 0
)
```

Arguments

`cat_names`	The names of the categorical variables
`cont_names`	The names of the continuous variables
`fill_strategy`	The strategy of filling
`add_col`	add_col
`fill_val`	fill_val

Value

None

Examples

```
## Not run:

procs = list(FillMissing(),Categorify(),Normalize())


## End(Not run)
```

FillStrategy_COMMON *COMMON*

Description

An enumeration.

Usage

```
FillStrategy_COMMON()
```

Value

None

FillStrategy_CONSTANT *CONSTANT*

Description

An enumeration.

Usage

```
FillStrategy_CONSTANT()
```

Value

None

FillStrategy_MEDIAN *MEDIAN*

Description

An enumeration.

Usage

```
FillStrategy_MEDIAN()
```

Value

None

find_coeffs *Find_coeffs*

Description

Find coefficients for warp tfm from ‘p1‘ to ‘p2‘

Usage

```
find_coeffs(p1, p2)
```

Arguments

p1	coefficient p1
p2	coefficient p2

Value

None

fine_tune *Fine_tune*

Description

Fine tune with ‘freeze‘ for ‘freeze_epochs‘ then with ‘unfreeze‘ from ‘epochs‘ using discriminative LR

Usage

```
fine_tune(
  object,
  epochs,
  base_lr = 0.002,
  freeze_epochs = 1,
  lr_mult = 100,
  pct_start = 0.3,
  div = 5,
  ...
)
```

Arguments

object	learner/model
epochs	epoch number
base_lr	base learning rate
freeze_epochs	freeze epochs number
lr_mult	learning rate multiply
pct_start	start percentage
div	divide
...	additional arguments

Value

None

---

fit.fastai.learner.Learner
Fit

Description

Fit the model on this learner with ‘lr‘ learning rate, ‘wd‘ weight decay for ‘epochs‘ with ‘callbacks‘ as cbs argument.

Usage

```
## S3 method for class 'fastai.learner.Learner'
fit(object, ...)
```

Arguments

object	a learner object
...	parameters to pass

Value

train history

`fit.fastai.tabular.learner.TabularLearner`
Fit

Description

Fit the model on this learner with 'lr' learning rate, 'wd' weight decay for 'epochs' with 'callbacks'.

Usage

```
## S3 method for class 'fastai.tabular.learner.TabularLearner'
fit(object, ...)
```

Arguments

`object`	model
`...`	additional arguments

Value

data frame

`fit.fastai.vision.gan.GANLearner`
Fit

Description

Fit the model on this learner with 'lr' learning rate, 'wd' weight decay for 'epochs' with 'callbacks'.

Usage

```
## S3 method for class 'fastai.vision.gan.GANLearner'
fit(object, ...)
```

Arguments

`object`	model
`...`	additonal parameters to pass

Value

train history

Examples

```
## Not run:

learn %>% fit(1, 2e-4, wd = 0)


## End(Not run)
```

`fit_flat_cos` *Fit_flat_cos*

Description

Fit_flat_cos

Usage

```
fit_flat_cos(
  object,
  n_epoch,
  lr = NULL,
  div_final = 1e+05,
  pct_start = 0.75,
  wd = NULL,
  cbs = NULL,
  reset_opt = FALSE
)
```

Arguments

`object`	learner/model
`n_epoch`	number of epochs
`lr`	learning rate
`div_final`	divide final value
`pct_start`	start percentage
`wd`	weight decay
`cbs`	callbacks
`reset_opt`	reset optimizer

Value

None

`fit_flat_lin` *Fit_flat_lin*

Description

Fit ‘self.model‘ for ‘n_epoch‘ at flat ‘start_lr‘ before ‘curve_type‘ annealing to ‘end_lr‘ with weight decay of ‘wd‘ and callbacks ‘cbs‘.

Usage

```
fit_flat_lin(
  object,
  n_epochs = 100,
  n_epochs_decay = 100,
  start_lr = NULL,
  end_lr = 0,
  curve_type = "linear",
  wd = NULL,
  cbs = NULL,
  reset_opt = FALSE
)
```

Arguments

`object`	model / learner
`n_epochs`	number of epochs
`n_epochs_decay`	number of epochs with decay
`start_lr`	Desired starting learning rate, used for beginning pct of training.
`end_lr`	Desired end learning rate, training will conclude at this learning rate.
`curve_type`	Curve type for learning rate annealing. Options are ’linear’, ’cosine’, and ’exponential’.
`wd`	weight decay
`cbs`	callbacks
`reset_opt`	reset optimizer

Value

None

fit_one_cycle *Fit one cycle*

Description

Fit one cycle

Usage

```
fit_one_cycle(object, ...)
```

Arguments

object	model
...	parameters to pass, e.g. lr, n_epoch, wd, and etc.

Value

None

fit_sgdr *Fit_sgdr*

Description

Fit_sgdr

Usage

```
fit_sgdr(
  object,
  n_cycles,
  cycle_len,
  lr_max = NULL,
  cycle_mult = 2,
  cbs = NULL,
  reset_opt = FALSE,
  wd = NULL
)
```

Arguments

object	learner/model
n_cycles	number of cycles
cycle_len	length of cycle
lr_max	maximum learning rate
cycle_mult	cycle mult
cbs	callbacks
reset_opt	reset optimizer
wd	weight decay

Value

None

FixedGANSwitcher *Fixed GAN Switcher*

Description

Switcher to do ‘n_crit‘ iterations of the critic then ‘n_gen‘ iterations of the generator.

Usage

```
FixedGANSwitcher(n_crit = 1, n_gen = 1)
```

Arguments

n_crit	number of discriminator
n_gen	number of generator

Value

None

fix_fit *Fix fit*

Description

Fix fit

Usage

```
fix_fit(disable_graph = FALSE)
```

Arguments

disable_graph	to remove dynamic plot, by default is FALSE

Value

None

fix_html *Fix_html*

Description

Various messy things we've seen in documents

Usage

```
fix_html(x)
```

Arguments

x	text

Value

string

Flatten *Flatten*

Description

Flatten ‘x‘ to a single dimension, e.g. at end of a model. ‘full‘ for rank-1 tensor

Usage

```
Flatten(full = FALSE)
```

Arguments

full	bool, full or not

flatten_check *Flatten check*

Description

Check that ‘out‘ and ‘targ‘ have the same number of elements and flatten them.

Usage

```
flatten_check(inp, targ)
```

Arguments

inp	predictions
targ	targets

Value

tensor

flatten_model *Flatten_model*

Description

Return the list of all submodules and parameters of ‘m‘

Usage

```
flatten_model(m)
```

Arguments

m	parameters

Value

None

Flip *Flip*

Description

Randomly flip a batch of images with a probability ‘p‘

Usage

```
Flip(
  p = 0.5,
  draw = NULL,
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  align_corners = TRUE,
  batch = FALSE
)
```

Arguments

p	probability
draw	draw
size	size of image
mode	mode
pad_mode	reflection, zeros, border as string parameter
align_corners	align corners ot not
batch	batch or not

Value

None

`FlipItem` *FlipItem*

Description

Randomly flip with probability ‘p‘

Usage

```
FlipItem(p = 0.5)
```

Arguments

p	probability

Value

None

`flip_mat` *Flip_mat*

Description

Return a random flip matrix

Usage

```
flip_mat(x, p = 0.5, draw = NULL, batch = FALSE)
```

Arguments

x	tensor
p	probability
draw	draw
batch	batch

Value

None

float *Tensor to float*

Description

Tensor to float

Usage

```
float(tensor)
```

Arguments

tensor	tensor

Value

tensor

floor.fastai.torch_core.TensorMask *Floor*

Description

Floor

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
floor(x)
```

Arguments

x	tensor

Value

tensor

floor_ *Floor*

Description

Floor

Usage

```
## S3 method for class 'torch.Tensor'
floor(x)
```

Arguments

x	tensor

Value

tensor

floor_div *Floor divide*

Description

Floor divide

Usage

```
## S3 method for class 'torch.Tensor'
x %/% y
```

Arguments

x	tensor
y	tensor

Value

tensor

floor_mod *Floor mod*

Description

Floor mod

Usage

```
## S3 method for class 'torch.Tensor'
x %% y
```

Arguments

x	tensor
y	tensor

Value

tensor

fmodule *Module*

Description

Module

Usage

```
fmodule(...)
```

Arguments

...	parameters to pass

Details

Decorator to create an nn$Module using f as forward method

Value

None

FolderDataset *FolderDataset*

Description

A PyTorch Dataset class that can be created from a folder ‘path‘ of images, for the sole purpose of inference. Optional ‘transforms‘

Usage

```
FolderDataset(path, transforms = NULL)
```

Arguments

path	path to dir
transforms	transformations

Details

can be provided. Attributes: ‘self.files‘: A list of the filenames in the folder. ‘self.totensor‘: ‘torchvision.transforms.ToTensor‘ transform. ‘self.transform‘: The transforms passed in as ‘transforms‘ to the constructor.

Value

None

force_plot *Force_plot*

Description

Visualizes the SHAP values with an added force layout. Accepts a class_id which is used to indicate the class of interest for a classification model.

Usage

```
force_plot(object, class_id = 0, ...)
```

Arguments

object	ShapInterpretation object
class_id	Accepts a class_id which is used to indicate the class of interest for a classification model. It can either be an int or str representation for a class of choice.
...	additional arguments

Value

None

`foreground_acc` *Foreground accuracy*

Description

Computes non-background accuracy for multiclass segmentation

Usage

```
foreground_acc(inp, targ, bkg_idx = 0, axis = 1)
```

Arguments

`inp`	predictions
`targ`	targets
`bkg_idx`	bkg_idx
`axis`	axis

Value

None

`ForgetMultGPU` *ForgetMultGPU*

Description

Wrapper around the CUDA kernels for the ForgetMult gate.

Usage

```
ForgetMultGPU(...)
```

Arguments

`...`	parameters to pass

Value

None

`forget_mult_CPU` *Forget_mult_CPU*

Description

ForgetMult gate applied to ‘x‘ and ‘f‘ on the CPU.

Usage

```
forget_mult_CPU(x, f, first_h = NULL, batch_first = TRUE, backward = FALSE)
```

Arguments

`x`	x
`f`	f
`first_h`	first_h
`batch_first`	batch_first
`backward`	backward

Value

None

`FuncSplitter` *FuncSplitter*

Description

Split ‘items‘ by result of ‘func‘ (‘TRUE‘ for validation, ‘FALSE‘ for training set).

Usage

```
FuncSplitter(func)
```

Arguments

`func`	function

Value

None

fView *View*

Description

Reshape x to size

Usage

```
fView(...)
```

Arguments

...	parameters to pass

Value

None

GANDiscriminativeLR *GAN Discriminative LR*

Description

'Callback' that handles multiplying the learning rate by 'mult_lr' for the critic.

Usage

```
GANDiscriminativeLR(mult_lr = 5)
```

Arguments

mult_lr	mult learning rate

GANLearner_from_learners *GAN Learner from learners*

Description

Create a GAN from ‘learn_gen‘ and ‘learn_crit‘.

Usage

```
GANLearner_from_learners(
  gen_learn,
  crit_learn,
  switcher = NULL,
  weights_gen = NULL,
  gen_first = FALSE,
  switch_eval = TRUE,
  show_img = TRUE,
  clip = NULL,
  cbs = NULL,
  metrics = NULL,
  loss_func = NULL,
  opt_func = Adam(),
  lr = 0.001,
  splitter = trainable_params(),
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE,
  moms = list(0.95, 0.85, 0.95)
)
```

Arguments

gen_learn	generator learner
crit_learn	discriminator learner
switcher	switcher
weights_gen	weights generator
gen_first	generator first
switch_eval	switch evaluation
show_img	show image or not
clip	clip value
cbs	Cbs is one or a list of Callbacks to pass to the Learner.

metrics	It is an optional list of metrics, that can be either functions or Metrics.
loss_func	loss function
opt_func	The function used to create the optimizer
lr	learning rate
splitter	It is a function that takes self.model and returns a list of parameter groups (or just one parameter group if there are no different parameter groups).
path	The folder where to work
model_dir	Path and model_dir are used to save and/or load models.
wd	It is the default weight decay used when training the model.
wd_bn_bias	It controls if weight decay is applied to BatchNorm layers and bias.
train_bn	It controls if BatchNorm layers are trained even when they are supposed to be frozen according to the splitter.
moms	The default momentums used in Learner$fit_one_cycle.

Value

None

`GANLearner_wgan` *Wgan*

Description

Create a WGAN from ‘data‘, ‘generator‘ and ‘critic‘.

Usage

```
GANLearner_wgan(
  dls,
  generator,
  critic,
  switcher = NULL,
  clip = 0.01,
  switch_eval = FALSE,
  gen_first = FALSE,
  show_img = TRUE,
  cbs = NULL,
  metrics = NULL,
  opt_func = Adam(),
  lr = 0.001,
  splitter = trainable_params,
  path = NULL,
  model_dir = "models",
  wd = NULL,
```

```
  wd_bn_bias = FALSE,
  train_bn = TRUE,
  moms = list(0.95, 0.85, 0.95)
)
```

Arguments

dls	dataloader
generator	generator
critic	critic
switcher	switcher
clip	clip value
switch_eval	switch evaluation
gen_first	generator first
show_img	show image or not
cbs	callbacks
metrics	metrics
opt_func	optimization function
lr	learning rate
splitter	splitter
path	path
model_dir	model directory
wd	weight decay
wd_bn_bias	weight decay bn bias
train_bn	It controls if BatchNorm layers are trained even when they are supposed to be frozen according to the splitter.
moms	momentums

Value

None

Examples

```
## Not run:

learn = GANLearner_wgan(dls, generator, critic, opt_func = partial(Adam(), mom=0.))


## End(Not run)
```

GANLoss *GAN Loss*

Description

Wrapper around ‘crit_loss_func‘ and ‘gen_loss_func‘

Usage

```
GANLoss(gen_loss_func, crit_loss_func, gan_model)
```

Arguments

gen_loss_func	generator loss funcion
crit_loss_func	discriminator loss function
gan_model	GAN model

Value

None

GANModule *GAN Module*

Description

Wrapper around a ‘generator‘ and a ‘critic‘ to create a GAN.

Usage

```
GANModule(generator = NULL, critic = NULL, gen_mode = FALSE)
```

Arguments

generator	generator
critic	critic
gen_mode	generator mode or not

Value

None

GANTrainer *GAN Trainer*

Description

Handles GAN Training.

Usage

```
GANTrainer(
  switch_eval = FALSE,
  clip = NULL,
  beta = 0.98,
  gen_first = FALSE,
  show_img = TRUE
)
```

Arguments

switch_eval	switch evaluation
clip	clip value
beta	beta parameter
gen_first	generator first
show_img	show image or not

Value

None

gan_critic *Gan critic*

Description

Critic to train a 'GAN'.

Usage

```
gan_critic(n_channels = 3, nf = 128, n_blocks = 3, p = 0.15)
```

Arguments

n_channels	number of channels
nf	number of features
n_blocks	number of blocks
p	probability

Value

GAN object

`gan_loss_from_func` *GAN loss from function*

Description

Define loss functions for a GAN from ‘loss_gen‘ and ‘loss_crit‘.

Usage

```
gan_loss_from_func(loss_gen, loss_crit, weights_gen = NULL)
```

Arguments

`loss_gen`	generator loss
`loss_crit`	discriminator loss
`weights_gen`	weight generator

Value

None

`GatherPredsCallback` *GatherPredsCallback*

Description

‘Callback‘ that saves the predictions and targets, optionally ‘with_loss‘

Usage

```
GatherPredsCallback(
  with_input = FALSE,
  with_loss = FALSE,
  save_preds = NULL,
  save_targs = NULL,
  concat_dim = 0
)
```

Arguments

with_input	include inputs or not
with_loss	include loss or not
save_preds	save predictions
save_targs	save targets/actuals
concat_dim	concatenate dimensions

Value

None

---

gauss_blur2d *Gauss_blur2d*

Description

Apply gaussian_blur2d kornia filter

Usage

```
gauss_blur2d(x, s)
```

Arguments

x	image
s	effect

Value

None

---

generate_noise *Generate noise*

Description

Generate noise

Usage

```
generate_noise(fn, size = 100)
```

Arguments

fn	path
size	the size

Value

None

Examples

```
## Not run:

generate_noise()


## End(Not run)
```

---

get_annotations *Get_annotations*

Description

Open a COCO style json in ‘fname‘ and returns the lists of filenames (with maybe ‘prefix‘) and labelled bboxes.

Usage

```
get_annotations(fname, prefix = NULL)
```

Arguments

fname	folder name
prefix	prefix

Value

None

get_audio_files *Get_audio_files*

Description

Get audio files in ‘path‘ recursively, only in ‘folders‘, if specified.

Usage

```
get_audio_files(path, recurse = TRUE, folders = NULL)
```

Arguments

path	path
recurse	recursive or not
folders	vector, folders

Value

None

get_bias *Get bias*

Description

Bias for item or user (based on ‘is_item‘) for all in ‘arr‘

Usage

```
get_bias(object, arr, is_item = TRUE, convert = TRUE)
```

Arguments

object	extract bias
arr	R data frame
is_item	logical, is item
convert	to R matrix

Value

tensor

Examples

```
## Not run:

movie_bias = learn %>% get_bias(top_movies, is_item = TRUE)


## End(Not run)
```

---

get_c *Get_c*

Description

Get_c

Usage

```
get_c(dls)
```

Arguments

dls	dataloader object

Value

number of layers

Examples

```
## Not run:

get_c(dls)


## End(Not run)
```

get_confusion_matrix *Extract confusion matrix*

Description

Extract confusion matrix

Usage

```
get_confusion_matrix(object)
```

Arguments

object	model

Value

matrix

Examples

```
## Not run:

model %>% get_confusion_matrix()


## End(Not run)
```

get_data_loaders *Get data loaders*

Description

Get data loaders

Usage

```
get_data_loaders(train_batch_size, val_batch_size)
```

Arguments

train_batch_size	train dataset batch size
val_batch_size	validation dataset batch size

Value

None

`get_dcm_matrix` *Get image matrix*

Description

Get image matrix

Usage

```
get_dcm_matrix(img, type = "raw", scan = "", size = 50, convert = TRUE)
```

Arguments

`img`	dicom file
`type`	img transformation
`scan`	apply uniform or gaussian blur effects
`size`	size of image
`convert`	to R matrix or keep tensor

Value

tensor

Examples

```
## Not run:

img = dcmread('hemorrhage.dcm')
img %>% get_dcm_matrix(type = 'raw')


## End(Not run)
```

`get_dicom_files` *get_dicom_files*

Description

Get dicom files in ‘path‘ recursively, only in ‘folders‘, if specified.

Usage

```
get_dicom_files(path, recurse = TRUE, folders = NULL)
```

Arguments

`path`	path to files
`recurse`	recursive or not
`folders`	folder names

Value

lsit of files

Examples

```
## Not run:

items = get_dicom_files("siim_small/train/")


## End(Not run)
```

`get_dls` *Get dls*

Description

Given image files from two domains (‘pathA‘, ‘pathB‘), create ‘DataLoaders‘ object.

Usage

```
get_dls(
  pathA,
  pathB,
  num_A = NULL,
  num_B = NULL,
  load_size = 512,
  crop_size = 256,
  bs = 4,
  num_workers = 2
)
```

Arguments

pathA	path A (from domain)
pathB	path B (to domain)
num_A	subset of A data
num_B	subset of B data
load_size	load size
crop_size	crop size
bs	bathc size
num_workers	number of workers

Details

Loading and randomly cropped sizes of ‘load_size‘ and ‘crop_size‘ are set to defaults of 512 and 256. Batch size is specified by ‘bs‘ (default=4).

Value

None

---

get_emb_sz *Get_emb_sz*

Description

Get default embedding size from ‘TabularPreprocessor‘ ‘proc‘ or the ones in ‘sz_dict‘

Usage

```
get_emb_sz(to, sz_dict = NULL)
```

Arguments

to	to
sz_dict	dictionary size

Value

None

`get_files` *Get_files*

Description

Get all the files in ‘path‘ with optional ‘extensions‘, optionally with ‘recurse‘, only in ‘folders‘, if specified.

Usage

```
get_files(
  path,
  extensions = NULL,
  recurse = TRUE,
  folders = NULL,
  followlinks = TRUE
)
```

Arguments

path	path
extensions	extensions
recurse	recurse
folders	folders
followlinks	followlinks

Value

list

`get_grid` *Get_grid*

Description

Return a grid of ‘n‘ axes, ‘rows‘ by ‘cols‘

Usage

```
get_grid(
  n,
  nrows = NULL,
  ncols = NULL,
  add_vert = 0,
  figsize = NULL,
  double = FALSE,
  title = NULL,
  return_fig = FALSE,
  imsize = 3
)
```

Arguments

`n`	n
`nrows`	number of rows
`ncols`	number of columns
`add_vert`	add vertical
`figsize`	figure size
`double`	double
`title`	title
`return_fig`	return figure or not
`imsize`	image size

Value

None

get_image_files *Get image files*

Description

Get image files in 'path' recursively, only in 'folders', if specified.

Usage

```
get_image_files(path, recurse = TRUE, folders = NULL)
```

Arguments

path	The folder where to work
recurse	recursive path
folders	folder names

Value

None

Examples

```
## Not run:

URLs_PETS()

path = 'oxford-iiit-pet'

path_img = 'oxford-iiit-pet/images'
fnames = get_image_files(path_img)


## End(Not run)
```

get_language_model *Get_language_model*

Description

Create a language model from 'arch' and its 'config'.

Usage

```
get_language_model(arch, vocab_sz, config = NULL, drop_mult = 1)
```

Arguments

arch	arch
vocab_sz	vocab_sz
config	config
drop_mult	drop_mult

Value

model

`get_preds_cyclegan` *Get_preds_cyclegan*

Description

A prediction function that takes the Learner object ‘learn‘ with the trained model, the ‘test_path‘ folder with the images to perform

Usage

```
get_preds_cyclegan(
  learn,
  test_path,
  pred_path,
  bs = 4,
  num_workers = 4,
  suffix = "tif"
)
```

Arguments

learn	learner/model
test_path	testdat path
pred_path	predict data path
bs	batch size
num_workers	number of workers
suffix	suffix

Details

batch inference on, and the output folder ‘pred_path‘ where the predictions will be saved, with a batch size ‘bs‘, ‘num_workers‘, and suffix of the prediction images ‘suffix‘ (default='png').

get_text_classifier *Get_text_classifier*

Description

Create a text classifier from ‘arch‘ and its ‘config‘, maybe ‘pretrained‘

Usage

```
get_text_classifier(
  arch,
  vocab_sz,
  n_class,
  seq_len = 72,
  config = NULL,
  drop_mult = 1,
  lin_ftrs = NULL,
  ps = NULL,
  pad_idx = 1,
  max_len = 1440,
  y_range = NULL
)
```

Arguments

arch	arch
vocab_sz	vocab_sz
n_class	n_class
seq_len	seq_len
config	config
drop_mult	drop_mult
lin_ftrs	lin_ftrs
ps	ps
pad_idx	pad_idx
max_len	max_len
y_range	y_range

Value

None

get_text_files *Get_text_files*

Description

Get text files in ‘path‘ recursively, only in ‘folders‘, if specified.

Usage

```
get_text_files(path, recurse = TRUE, folders = NULL)
```

Arguments

path	path
recurse	recurse
folders	folders

Value

None

get_weights *Get weights*

Description

Weight for item or user (based on ‘is_item‘) for all in ‘arr‘

Usage

```
get_weights(object, arr, is_item = TRUE, convert = FALSE)
```

Arguments

object	extract weights
arr	R data frame
is_item	logical, is item
convert	to R matrix

Value

tensor

Examples

```
## Not run:

movie_w = learn %>% get_weights(top_movies, is_item = TRUE, convert = TRUE)


## End(Not run)
```

`GrandparentSplitter` *GrandparentSplitter*

Description

Split ‘items‘ from the grand parent folder names (‘train_name‘ and ‘valid_name‘).

Usage

```
GrandparentSplitter(train_name = "train", valid_name = "valid")
```

Arguments

`train_name`	train folder name
`valid_name`	validation folder name

Value

None

`grayscale` *Grayscale*

Description

Tensor to grayscale tensor. Uses the ITU-R 601-2 luma transform.

Usage

```
grayscale(x)
```

Arguments

`x`	tensor

Value

None

greater *Greater*

Description

Greater

Usage

```
## S3 method for class 'torch.Tensor'
a > b
```

Arguments

a	tensor
b	tensor

Value

tensor

greater_or_equal *Greater or equal*

Description

Greater or equal

Usage

```
## S3 method for class 'torch.Tensor'
a >= b
```

Arguments

a	tensor
b	tensor

Value

tensor

HammingLoss *HammingLoss*

Description

Hamming loss for single-label classification problems

Hamming loss for single-label classification problems

Usage

```
HammingLoss(axis = -1, sample_weight = NULL)

HammingLoss(axis = -1, sample_weight = NULL)
```

Arguments

axis	axis
sample_weight	sample_weight

Value

Loss object

None

HammingLossMulti *HammingLossMulti*

Description

Hamming loss for multi-label classification problems

Usage

```
HammingLossMulti(
  thresh = 0.5,
  sigmoid = TRUE,
  labels = NULL,
  sample_weight = NULL
)
```

Arguments

thresh	threshold
sigmoid	sigmoid
labels	labels
sample_weight	sample_weight

Value

Loss object

`has_pool_type` *Has_pool_type*

Description

Return ‘TRUE‘ if ‘m‘ is a pooling layer or has one in its children

Usage

```
has_pool_type(m)
```

Arguments

`m`	parameters

Value

None

`HookCallback` *HookCallback*

Description

‘Callback‘ that can be used to register hooks on ‘modules‘

Usage

```
HookCallback(
  modules = NULL,
  every = NULL,
  remove_end = TRUE,
  is_forward = TRUE,
  detach = TRUE,
  cpu = TRUE
)
```

Arguments

modules	the modules
every	int, every epoch
remove_end	logical, remove_end
is_forward	logical, is_forward
detach	detach
cpu	to cpu or not

Value

None

---

hsv2rgb *Hsv2rgb*

Description

Converts a HSV image to an RGB image.

Usage

```
hsv2rgb(img)
```

Arguments

img	image object

Value

None

Hue *Hue*

Description

Apply change in hue of ‘max_hue‘ to batch of images with probability ‘p‘.

Usage

```
Hue(max_hue = 0.1, p = 0.75, draw = NULL, batch = FALSE)
```

Arguments

max_hue	maximum hue
p	probability
draw	draw
batch	batch

Value

None

---

hug *Transformer module*

Description

Transformer module

Usage

```
hug()
```

Value

None

---

icnr_init *Icnr_init*

Description

ICNR init of ‘x‘, with ‘scale‘ and ‘init‘ function

Usage

```
icnr_init(x, scale = 2, init = nn$init$kaiming_normal_)
```

Arguments

x	tensor
scale	int, scale
init	initializer

Value

None

Image *Image*

Description

Image

Usage

```
Image(...)
```

Arguments

...	parameters to pass

Value

None

image2tensor *Image2tensor*

Description

Transform image to byte tensor in ‘c*h*w‘ dim order.

Usage

```
image2tensor(img)
```

Arguments

img	image

Value

None

`ImageBlock` *ImageBlock*

Description

A 'TransformBlock' for images of 'cls'

Usage

```
ImageBlock(...)
```

Arguments

...	parameters to pass

Value

block

`ImageBW_create` *ImageBW_create*

Description

Open an 'Image' from path 'fn'

Usage

```
ImageBW_create(fn)
```

Arguments

fn	file name

Value

None

`ImageDataLoaders_from_csv` *ImageDataLoaders from csv*

Description

Create from ‘path/csv_fname‘ using ‘fn_col‘ and ‘label_col‘

Usage

```
ImageDataLoaders_from_csv(
  path,
  csv_fname = "labels.csv",
  header = "infer",
  delimiter = NULL,
  valid_pct = 0.2,
  seed = NULL,
  fn_col = 0,
  folder = NULL,
  suff = "",
  label_col = 1,
  label_delim = NULL,
  y_block = NULL,
  valid_col = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  bs = 64,
  val_bs = NULL,
  size = NULL,
  shuffle_train = TRUE,
  device = NULL,
  ...
)
```

Arguments

`path`	The folder where to work
`csv_fname`	csv file name
`header`	header
`delimiter`	delimiter
`valid_pct`	validation percentage
`seed`	random seed
`fn_col`	column name
`folder`	folder name

suff	suff
label_col	label column
label_delim	label delimiter
y_block	y_block
valid_col	validation column
item_tfms	One or several transforms applied to the items before batching them
batch_tfms	One or several transforms applied to the batches once they are formed
bs	batch size
val_bs	The batch size for the validation DataLoader (defaults to bs)
size	image size
shuffle_train	If we shuffle the training DataLoader or not
device	device name
...	additional parameters to pass

Value

None

`ImageDataLoaders_from_dblock` *ImageDataLoaders from dblock*

Description

Create a dataloaders from a given ‘dblock‘

Usage

```
ImageDataLoaders_from_dblock(
  dblock,
  source,
  path = ".",
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL,
  ...
)
```

Arguments

dblock	dblock
source	source folder
path	The folder where to work
bs	batch size
val_bs	The batch size for the validation DataLoader (defaults to bs)
shuffle_train	If we shuffle the training DataLoader or not
device	device name
...	additional parameters to pass

Value

None

`ImageDataLoaders_from_df`

ImageDataLoaders from df

Description

Create from ‘df‘ using ‘fn_col‘ and ‘label_col‘

Usage

```
ImageDataLoaders_from_df(
  df,
  path = ".",
  valid_pct = 0.2,
  seed = NULL,
  fn_col = 0,
  folder = NULL,
  suff = "",
  label_col = 1,
  label_delim = NULL,
  y_block = NULL,
  valid_col = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL,
  ...
)
```

Arguments

df	data frame
path	The folder where to work
valid_pct	validation percentage
seed	random seed
fn_col	column name
folder	folder name
suff	suff
label_col	label column
label_delim	label separator
y_block	y_block
valid_col	validation column
item_tfms	One or several transforms applied to the items before batching them
batch_tfms	One or several transforms applied to the batches once they are formed
bs	batch size
val_bs	The batch size for the validation DataLoader (defaults to bs)
shuffle_train	shuffle_train
device	device
...	additional parameters to pass

Value

None

ImageDataLoaders_from_folder

ImageDataLoaders from folder

Description

Create from imagenet style dataset in ‘path‘ with ‘train‘ and ‘valid‘ subfolders (or provide ‘valid_pct‘)

Usage

```
ImageDataLoaders_from_folder(
  path,
  train = "train",
  valid = "valid",
  valid_pct = NULL,
  seed = NULL,
  vocab = NULL,
```

```
  item_tfms = NULL,
  batch_tfms = NULL,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL,
  size = NULL,
  ...
)
```

Arguments

path	The folder where to work
train	train data
valid	validation data
valid_pct	validion percentage
seed	random seed
vocab	vocabulary
item_tfms	One or several transforms applied to the items before batching them
batch_tfms	One or several transforms applied to the batches once they are formed
bs	batch size
val_bs	The batch size for the validation DataLoader (defaults to bs)
shuffle_train	If we shuffle the training DataLoader or not
device	device name
size	image size
...	additional parameters to pass

ImageDataLoaders_from_lists

ImageDataLoaders from lists

Description

Create from list of ‘fnames‘ and ‘labels‘ in ‘path‘

Usage

```
ImageDataLoaders_from_lists(
  path,
  fnames,
  labels,
  valid_pct = 0.2,
  seed = NULL,
```

```
  y_block = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL,
  ...
)
```

Arguments

path	The folder where to work
fnames	file names
labels	labels
valid_pct	validation percentage
seed	random seed
y_block	y_block
item_tfms	One or several transforms applied to the items before batching them
batch_tfms	One or several transforms applied to the batches once they are formed
bs	batch size
val_bs	The batch size for the validation DataLoader (defaults to bs)
shuffle_train	If we shuffle the training DataLoader or not
device	device name
...	additional parameters to pass

Value

None

ImageDataLoaders_from_name_re

ImageDataLoaders from name regex

Description

Create from the name attrs of ‘fnames‘ in ‘path‘s with re expression ‘pat‘

Usage

```
ImageDataLoaders_from_name_re(
  path,
  fnames,
  pat,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  ...
)
```

Arguments

path	The folder where to work
fnames	folder names
pat	an argument that requires regex
bs	The batch size
val_bs	The batch size for the validation DataLoader (defaults to bs)
shuffle_train	If we shuffle the training DataLoader or not
device	device name
item_tfms	One or several transforms applied to the items before batching them
batch_tfms	One or several transforms applied to the batches once they are formed
...	additional parameters to pass

Value

None

Examples

```
## Not run:

URLs_PETS()

path = 'oxford-iiit-pet'

dls = ImageDataLoaders_from_name_re(
path, fnames, pat='(.+)_\\d+.jpg$',
item_tfms = RandomResizedCrop(460, min_scale=0.75), bs = 10,
batch_tfms = list(aug_transforms(size = 299, max_warp = 0),
                  Normalize_from_stats( imagenet_stats() )
),
device = 'cuda'
```

```
)


## End(Not run)
```

`ImageDataLoaders_from_path_func` *ImageDataLoaders from path function*

Description

Create from list of ‘fnames‘ in ‘path‘s with ‘label_func‘

Usage

```
ImageDataLoaders_from_path_func(
  path,
  fnames,
  label_func,
  valid_pct = 0.2,
  seed = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL,
  ...
)
```

Arguments

`path`	The folder where to work
`fnames`	file names
`label_func`	label function
`valid_pct`	The random percentage of the dataset to set aside for validation (with an optional seed)
`seed`	random seed
`item_tfms`	One or several transforms applied to the items before batching them
`batch_tfms`	One or several transforms applied to the batches once they are formed
`bs`	batch size
`val_bs`	The batch size for the validation DataLoader (defaults to bs)
`shuffle_train`	If we shuffle the training DataLoader or not
`device`	device name
`...`	additional parameters to pass

Value

None

`ImageDataLoaders_from_path_re` *ImageDataLoaders from path re*

Description

Create from list of ‘fnames‘ in ‘path‘s with re expression ‘pat‘

Usage

```
ImageDataLoaders_from_path_re(
  path,
  fnames,
  pat,
  valid_pct = 0.2,
  seed = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL,
  ...
)
```

Arguments

`path`	The folder where to work
`fnames`	file names
`pat`	an argument that requires regex
`valid_pct`	The random percentage of the dataset to set aside for validation (with an optional seed)
`seed`	random seed
`item_tfms`	One or several transforms applied to the items before batching them
`batch_tfms`	One or several transforms applied to the batches once they are formed
`bs`	batch size
`val_bs`	The batch size for the validation DataLoader (defaults to bs)
`shuffle_train`	If we shuffle the training DataLoader or not
`device`	device name
`...`	additional parameters to pass

Value

None

`imagenet_stats` *Imagenet statistics*

Description

Imagenet statistics

Usage

```
imagenet_stats()
```

Value

vector

Examples

```
## Not run:

imagenet_stats()


## End(Not run)
```

`Image_create` *Image_create*

Description

Open an ‘Image‘ from path ‘fn‘

Usage

```
Image_create(fn)
```

Arguments

`fn`	file name

Value

None

Image_open *Image_open*

Description

Opens and identifies the given image file.

Usage

```
Image_open(fp, mode = "r")
```

Arguments

fp	fp
mode	mode

Value

None

Image_resize *Resize*

Description

Returns a resized copy of this image.

Usage

```
Image_resize(img, size, resample = 3, box = NULL, reducing_gap = NULL)
```

Arguments

img	image
size	size
resample	resample
box	box
reducing_gap	reducing_gap

Value

None

InceptionModule *InceptionModule*

Description

The inception Module from ‘ni‘ inputs to len(’kss’)*‘nb_filters‘+‘bottleneck_size‘

Usage

```
InceptionModule(
  ni,
  nb_filters = 32,
  kss = c(39, 19, 9),
  bottleneck_size = 32,
  stride = 1
)
```

Arguments

ni	number of input channels
nb_filters	the number of filters
kss	kernel size
bottleneck_size	bottleneck size
stride	stride

Value

module

IndexSplitter *Index Splitter*

Description

Split ‘items‘ so that ‘val_idx‘ are in the validation set and the others in the training set

Usage

```
IndexSplitter(valid_idx)
```

Arguments

valid_idx	The indices to use for the validation set (defaults to a random split otherwise)

Value

None

`init` *Wandb init*

Description

Initialize a wandb Run.

Usage

```
init(...)
```

Arguments

`...`	parameters to pass

Value

wandb Run object

None

see https

//docs.wandb.com/library/init

`init_default` *Init_default*

Description

Initialize ‘m‘ weights with ‘func‘ and set ‘bias‘ to 0.

Usage

```
init_default(m, func = nn$init$kaiming_normal_)
```

Arguments

`m`	parameters
`func`	function

Value

None

init_linear *Init_linear*

Description

Init_linear

Usage

```
init_linear(m, act_func = NULL, init = "auto", bias_std = 0.01)
```

Arguments

m	parameter
act_func	activation function
init	initializer
bias_std	bias standard deviation

Value

None

install_fastai *Install fastai*

Description

Install fastai

Usage

```
install_fastai(
  version,
  gpu = FALSE,
  cuda_version = "10.1",
  overwrite = FALSE,
 extra_pkgs = c("kaggle", "transformers", "pytorch_lightning", "timm", "catalyst",
    "ignite", "tensorboard", "fastinference", "shap")
)
```

Arguments

version	specify version
gpu	installation of gpu
cuda_version	if gpu true, then cuda version is required. By default it is 10.1
overwrite	will install all the dependencies
extra_pkgs	character vector of additional packages

Value

None

InstanceNorm *InstanceNorm*

Description

InstanceNorm layer with ‘nf‘ features and ‘ndim‘ initialized depending on ‘norm_type‘.

Usage

```
InstanceNorm(
  nf,
  ndim = 2,
  norm_type = 5,
  affine = TRUE,
  eps = 1e-05,
  momentum = 0.1,
  track_running_stats = FALSE
)
```

Arguments

nf	input shape
ndim	dimension number
norm_type	normalization type
affine	affine
eps	epsilon
momentum	momentum
track_running_stats	track running statistics

Value

None

IntToFloatTensor *IntToFloatTensor*

Description

Transform image to float tensor, optionally dividing by 255 (e.g. for images).

Usage

```
IntToFloatTensor(div = 255, div_mask = 1)
```

Arguments

div	divide value
div_mask	divide mask

Value

None

InvisibleTensor *Invisible Tensor*

Description

Invisible Tensor

Usage

```
InvisibleTensor(x)
```

Arguments

x	tensor

Value

None

in_channels *In_channels*

Description

Return the shape of the first weight layer in ‘m‘.

Usage

```
in_channels(m)
```

Arguments

m	parameters

Value

None

is_rmarkdown *Is Rmarkdown?*

Description

Is Rmarkdown?

Usage

```
is_rmarkdown()
```

Value

logical True/False

Jaccard *Jaccard*

Description

Jaccard score for single-label classification problems

Usage

```
Jaccard(
  axis = -1,
  labels = NULL,
  pos_label = 1,
  average = "binary",
  sample_weight = NULL
)
```

Arguments

axis	axis
labels	labels
pos_label	pos_label
average	average
sample_weight	sample_weight

Value

None

JaccardCoeff *JaccardCoeff*

Description

Implementation of the Jaccard coefficient that is lighter in RAM

Usage

```
JaccardCoeff(axis = 1)
```

Arguments

axis	axis

Value

None

JaccardMulti *JaccardMulti*

Description

Jaccard score for multi-label classification problems

Usage

```
JaccardMulti(
  thresh = 0.5,
  sigmoid = TRUE,
  labels = NULL,
  pos_label = 1,
  average = "macro",
  sample_weight = NULL
)
```

Arguments

thresh	thresh
sigmoid	sigmoid
labels	labels
pos_label	pos_label
average	average
sample_weight	sample_weight

Value

None

kg *Kaggle module*

Description

Kaggle module

Usage

```
kg()
```

Value

None

L *L*

Description

Behaves like a list of ‘items‘ but can also index with list of indices or masks

Usage

```
L(...)
```

Arguments

...	arguments to pass

L1LossFlat *L1LossFlat*

Description

Flattens input and output, same as nn$L1LossFlat

Usage

```
L1LossFlat(...)
```

Arguments

...	parameters to pass

Value

Loss object

l2_reg *L2_reg*

Description

L2 regularization as adding ‘wd*p‘ to ‘p$grad‘

Usage

```
l2_reg(p, lr, wd, do_wd = TRUE, ...)
```

Arguments

p	p
lr	learning rate
wd	weight decay
do_wd	do_wd
...	additional arguments to pass

Value

None

Examples

```
## Not run:

tst_param = function(val, grad = NULL) {
  "Create a tensor with `val` and a gradient of `grad` for testing"
  res = tensor(val) %>% float()

  if(is.null(grad)) {
    grad = tensor(val / 10)
  } else {
    grad = tensor(grad)
  }

  res$grad = grad %>% float()
  res
}
p = tst_param(1., 0.1)
l2_reg(p, 1., 0.1)


## End(Not run)
```

LabeledBBox *LabeledBBox*

Description

Basic type for a list of bounding boxes in an image

Usage

```
LabeledBBox(...)
```

Arguments

...	parameters to pass

Value

None

LabelSmoothingCrossEntropy *LabelSmoothingCrossEntropy*

Description

Same as ‘nn$Module‘, but no need for subclasses to call ‘super()$__init__‘

Usage

```
LabelSmoothingCrossEntropy(eps = 0.1, reduction = "mean")
```

Arguments

eps	epsilon
reduction	reduction, defaults to mean

Value

Loss object

LabelSmoothingCrossEntropyFlat *LabelSmoothingCrossEntropyFlat*

Description

Same as ‘nn$Module‘, but no need for subclasses to call ‘super().__init__‘

Usage

```
LabelSmoothingCrossEntropyFlat(...)
```

Arguments

...	parameters to pass

Value

Loss object

Lamb *Lamb*

Description

Lamb

Usage

```
Lamb(...)
```

Arguments

...	parameters to pass

Value

None

Lambda *Lambda*

Description

An easy way to create a pytorch layer for a simple ‘func‘

Usage

```
Lambda(func)
```

Arguments

func	function

Value

None

lamb_step *Lamb_step*

Description

Step for LAMB with ‘lr‘ on ‘p‘

Usage

```
lamb_step(p, lr, mom, step, sqr_mom, grad_avg, sqr_avg, eps, ...)
```

Arguments

p	p
lr	learning rate
mom	momentum
step	step
sqr_mom	sqr momentum
grad_avg	gradient average
sqr_avg	sqr average
eps	epsilon
...	additional arguments to pass

Value

None

language_model_learner *Language_model_learner*

Description

Create a ‘Learner‘ with a language model from ‘dls‘ and ‘arch‘.

Usage

```
language_model_learner(
  dls,
  arch,
  config = NULL,
  drop_mult = 1,
  backwards = FALSE,
  pretrained = TRUE,
  pretrained_fnames = NULL,
  opt_func = Adam(),
  lr = 0.001,
  cbs = NULL,
  metrics = NULL,
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE,
  moms = list(0.95, 0.85, 0.95),
  ...
)
```

Arguments

dls	dls
arch	arch
config	config
drop_mult	drop_mult
backwards	backwards
pretrained	pretrained
pretrained_fnames	pretrained_fnames
opt_func	opt_func
lr	lr
cbs	cbs

metrics	metrics
path	path
model_dir	model_dir
wd	wd
wd_bn_bias	wd_bn_bias
train_bn	train_bn
moms	moms
...	additional arguments

Value

None

Larc *Larc*

Description

Larc

Usage

```
Larc(...)
```

Arguments

...	parameters to pass

Value

None

larc_layer_lr *Larc_layer_lr*

Description

Computes the local lr before weight decay is applied

Usage

```
larc_layer_lr(p, lr, trust_coeff, wd, eps, clip = TRUE, ...)
```

Arguments

p	p
lr	learning rate
trust_coeff	trust_coeff
wd	weight decay
eps	epsilon
clip	clip
...	additional arguments to pass

Value

None

larc_step *Larc_step*

Description

Step for LARC ‘local_lr‘ on ‘p‘

Usage

```
larc_step(p, local_lr, grad_avg = NULL, ...)
```

Arguments

p	p
local_lr	local learning rate
grad_avg	gradient average
...	additional args to pass

Value

None

Learner *Learner*

Description

Learner

Usage

```
Learner(...)
```

Arguments

...	parameters to pass

Value

None

Examples

```
## Not run:

model = LitModel()

data = Data_Loaders(model$train_dataloader(), model$val_dataloader())$cuda()

learn = Learner(data, model, loss_func = F$cross_entropy, opt_func = Adam,
                metrics = accuracy)


## End(Not run)
```

length *Length*

Description

Length

Usage

```
## S3 method for class 'torch.Tensor'
length(x)
```

Arguments

x	tensor

Value

tensor

---

length.fastai.torch_core.TensorMask *Length*

Description

Length

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
length(x)
```

Arguments

x	tensor

Value

tensor

---

less *Less*

Description

Less

Usage

```
## S3 method for class 'torch.Tensor'
a < b
```

Arguments

a	tensor
b	tensor

Value

tensor

less_or_equal *Less or equal*

Description

Less or equal

Usage

```
## S3 method for class 'torch.Tensor'
a <= b
```

Arguments

a	tensor
b	tensor

Value

tensor

LightingTfm *LightingTfm*

Description

Apply ‘fs‘ to the logits

Usage

```
LightingTfm(fs, ...)
```

Arguments

fs	fs
...	parameters to pass

Value

None

LinBnDrop *LinBnDrop*

Description

Module grouping 'BatchNorm1d', 'Dropout' and 'Linear' layers

Usage

```
LinBnDrop(n_in, n_out, bn = TRUE, p = 0, act = NULL, lin_first = FALSE)
```

Arguments

n_in	input shape
n_out	output shape
bn	bn
p	probability
act	activation
lin_first	linear first

Value

None

LinearDecoder *LinearDecoder*

Description

To go on top of a RNNCore module and create a Language Model.

Usage

```
LinearDecoder(n_out, n_hid, output_p = 0.1, tie_encoder = NULL, bias = TRUE)
```

Arguments

n_out	n_out
n_hid	n_hid
output_p	output_p
tie_encoder	tie_encoder
bias	bias

Value

None

LitModel *Lit Model*

Description

Lit Model

Usage

```
LitModel()
```

Value

model

LMDataLoader *LMDataLoader*

Description

A 'DataLoader' suitable for language modeling

Usage

```
LMDataLoader(
  dataset,
  lens = NULL,
  cache = 2,
  bs = 64,
  seq_len = 72,
  num_workers = 0,
  shuffle = FALSE,
  verbose = FALSE,
  do_setup = TRUE,
  pin_memory = FALSE,
  timeout = 0L,
  batch_size = NULL,
  drop_last = FALSE,
  indexed = NULL,
  n = NULL,
  device = NULL
)
```

Arguments

dataset	dataset
lens	lens
cache	cache
bs	bs
seq_len	seq_len
num_workers	num_workers
shuffle	shuffle
verbose	verbose
do_setup	do_setup
pin_memory	pin_memory
timeout	timeout
batch_size	batch_size
drop_last	drop_last
indexed	indexed
n	n
device	device

Value

text loader

LMLearner *LMLearner*

Description

Add functionality to 'TextLearner' when dealingwith a language model

Add functionality to 'TextLearner' when dealing with a language model

Usage

```
LMLearner(
  dls,
  model,
  alpha = 2,
  beta = 1,
  moms = list(0.8, 0.7, 0.8),
  loss_func = NULL,
  opt_func = Adam(),
  lr = 0.001,
  splitter = trainable_params(),
```

```
  cbs = NULL,
  metrics = NULL,
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE
)

LMLearner(
  dls,
  model,
  alpha = 2,
  beta = 1,
  moms = list(0.8, 0.7, 0.8),
  loss_func = NULL,
  opt_func = Adam(),
  lr = 0.001,
  splitter = trainable_params(),
  cbs = NULL,
  metrics = NULL,
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE
)
```

Arguments

dls	dls
model	model
alpha	alpha
beta	beta
moms	moms
loss_func	loss_func
opt_func	opt_func
lr	lr
splitter	splitter
cbs	cbs
metrics	metrics
path	path
model_dir	model_dir
wd	wd
wd_bn_bias	wd_bn_bias
train_bn	train_bn

Value

text loader

None

`LMLearner_predict` *LMLearner_predict*

Description

Return ‘text‘ and the ‘n_words‘ that come after

Usage

```
LMLearner_predict(
  text,
  n_words = 1,
  no_unk = TRUE,
  temperature = 1,
  min_p = NULL,
  no_bar = FALSE,
  decoder = decode_spec_tokens(),
  only_last_word = FALSE
)
```

Arguments

`text`	text
`n_words`	n_words
`no_unk`	no_unk
`temperature`	temperature
`min_p`	min_p
`no_bar`	no_bar
`decoder`	decoder
`only_last_word`	only_last_word

Value

None

loaders *Loaders*

Description

a loader from Catalyst

Usage

```
loaders()
```

Value

None

Examples

```
## Not run:

# trigger download
loaders()


## End(Not run)
```

load_dataset *Load_dataset*

Description

A helper function for getting a DataLoader for images in the folder ‘test_path‘, with batch size ‘bs‘, and number of workers ‘num_workers‘

Usage

```
load_dataset(test_path, bs = 4, num_workers = 4)
```

Arguments

test_path	test path (directory)
bs	batch size
num_workers	number of workers

Value

None

load_ignore_keys *Load_ignore_keys*

Description

Load ‘wgts‘ in ‘model‘ ignoring the names of the keys, just taking parameters in order

Usage

```
load_ignore_keys(model, wgts)
```

Arguments

model	model
wgts	wgts

Value

None

load_image *Load_image*

Description

Open and load a ‘PIL.Image‘ and convert to ‘mode‘

Usage

```
load_image(fn, mode = NULL)
```

Arguments

fn	file name
mode	mode

Value

None

load_model_text *Load_model_text*

Description

Load ‘model‘ from ‘file‘ along with ‘opt‘ (if available, and if ‘with_opt‘)

Usage

```
load_model_text(
  file,
  model,
  opt,
  with_opt = NULL,
  device = NULL,
  strict = TRUE
)
```

Arguments

file	file
model	model
opt	opt
with_opt	with_opt
device	device
strict	strict

Value

None

load_pre_models *Timm models*

Description

Timm models

Usage

```
load_pre_models()
```

Value

None

load_tokenized_csv *Load_tokenized_csv*

Description

Utility function to quickly load a tokenized csv and the corresponding counter

Usage

```
load_tokenized_csv(fname)
```

Arguments

fname	file name

Value

None

log *Log*

Description

Log

Usage

```
## S3 method for class 'torch.Tensor'
log(x, base = exp(1))
```

Arguments

x	tensor
base	base parameter

Value

tensor

log.fastai.torch_core.TensorMask *Log*

Description

Log

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
log(x, base = exp(1))
```

Arguments

x	tensor
base	base parameter

Value

tensor

log1p *Log1p*

Description

Log1p

Usage

```
## S3 method for class 'torch.Tensor'
log1p(x)
```

Arguments

x	tensor

Value

tensor

log1p.fastai.torch_core.TensorMask *Log1p*

Description

Log1p

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
log1p(x)
```

Arguments

x	tensor

Value

tensor

logical_and *Logical_and*

Description

Logical_and

Usage

```
## S3 method for class 'torch.Tensor'
x & y
```

Arguments

x	tensor
y	tensor

Value

tensor

logical_not_ *Logical_not*

Description

Logical_not

Usage

```
## S3 method for class 'torch.Tensor'
!x
```

Arguments

x	tensor

Value

tensor

logical_or *Logical_or*

Description

Logical_or

Usage

```
## S3 method for class 'torch.Tensor'
x | y
```

Arguments

x	tensor
y	tensor

Value

tensor

`login` *Wandb login*

Description

Log in to W&B.

Usage

```
login(anonymous = NULL, key = NULL, relogin = NULL, host = NULL, force = NULL)
```

Arguments

`anonymous`	must,never,allow,false,true
`key`	API key (secret)
`relogin`	relogin or not
`host`	host address
`force`	whether to force a user to be logged into wandb when running a script

Value

None

`Lookahead` *Lookahead*

Description

Lookahead

Usage

```
Lookahead(...)
```

Arguments

`...`	parameters to pass

Value

None

LossMetric *LossMetric*

Description

Create a metric from ‘loss_func.attr‘ named ‘nm‘

Usage

```
LossMetric(attr, nm = NULL)
```

Arguments

attr	attr
nm	nm

Value

None

lr_find *Lr_find*

Description

Launch a mock training to find a good learning rate, return lr_min, lr_steep if ‘suggestions‘ is TRUE

Usage

```
lr_find(
  object,
  start_lr = 1e-07,
  end_lr = 10,
  num_it = 100,
  stop_div = TRUE,
  suggestions = TRUE,
  ...
)
```

Arguments

object	learner
start_lr	starting learning rate
end_lr	end learning rate
num_it	number of iterations
stop_div	stop div or not
suggestions	suggestions
...	additional arguments to pass

Value

data frame

Examples

```
## Not run:

model %>% lr_find()
model %>% plot_lr_find(dpi = 200)


## End(Not run)
```

---

mae *MAE*

Description

Mean absolute error between ‘inp‘ and ‘targ‘.

Usage

```
mae(inp, targ)
```

Arguments

inp	predictions
targ	targets

Value

None

make_vocab *Make_vocab*

Description

Create a vocab of ‘max_vocab‘ size from ‘Counter‘ ‘count‘ with items present more than ‘min_freq‘

Usage

```
make_vocab(count, min_freq = 3, max_vocab = 60000, special_toks = NULL)
```

Arguments

count	count
min_freq	min_freq
max_vocab	max_vocab
special_toks	special_toks

Value

None

mask2bbox *Mask2bbox*

Description

Mask2bbox

Usage

```
mask2bbox(mask, convert = TRUE)
```

Arguments

mask	mask
convert	to R matrix

Value

tensor

MaskBlock *MaskBlock*

Description

A 'TransformBlock' for segmentation masks, potentially with 'codes'

Usage

```
MaskBlock(codes = NULL)
```

Arguments

codes	codes

Value

block

masked_concat_pool *Masked_concat_pool*

Description

Pool 'MultiBatchEncoder' outputs into one vector [last_hidden, max_pool, avg_pool]

Usage

```
masked_concat_pool(output, mask, bptt)
```

Arguments

output	output
mask	mask
bptt	bptt

Value

None

MaskFreq *Mask Freq*

Description

Google SpecAugment frequency masking from https://arxiv.org/abs/1904.08779.

Usage

```
MaskFreq(num_masks = 1, size = 20, start = NULL, val = NULL)
```

Arguments

num_masks	number of masks
size	size
start	starting point
val	value

Value

None

MaskTime *MaskTime*

Description

Google SpecAugment time masking from https://arxiv.org/abs/1904.08779.

Usage

```
MaskTime(num_masks = 1, size = 20, start = NULL, val = NULL)
```

Arguments

num_masks	number of masks
size	size
start	starting point
val	value

Value

None

Mask_create *Mask_create*

Description

Delegates (‘__call__‘,‘decode‘,‘setup‘) to (‘encodes‘,‘decodes‘,‘setups‘) if ‘split_idx‘ matches

Usage

```
Mask_create(enc = NULL, dec = NULL, split_idx = NULL, order = NULL)
```

Arguments

enc	encoder
dec	decoder
split_idx	split by index
order	order

Value

None

mask_from_blur *Mask from blur*

Description

Mask from blur

Usage

```
mask_from_blur(img, window, sigma = 0.3, thresh = 0.05, remove_max = TRUE)
```

Arguments

img	image
window	windowing effect
sigma	sigma
thresh	thresholf point
remove_max	remove maximum or not

mask_tensor *Mask_tensor*

Description

Mask elements of ‘x‘ with ‘neutral‘ with probability ‘1-p‘

Usage

```
mask_tensor(x, p = 0.5, neutral = 0, batch = FALSE)
```

Arguments

x	tensor
p	probability
neutral	neutral
batch	batch

Value

None

match_embeds *Match_embeds*

Description

Convert the embedding in ‘old_wgts‘ to go from ‘old_vocab‘ to ‘new_vocab‘.

Usage

```
match_embeds(old_wgts, old_vocab, new_vocab)
```

Arguments

old_wgts	old_wgts
old_vocab	old_vocab
new_vocab	new_vocab

Value

None

MatthewsCorrCoef *MatthewsCorrCoef*

Description

Matthews correlation coefficient for single-label classification problems

Usage

```
MatthewsCorrCoef(...)
```

Arguments

...	parameters to pass

Value

None

MatthewsCorrCoefMulti *MatthewsCorrCoefMulti*

Description

Matthews correlation coefficient for multi-label classification problems

Usage

```
MatthewsCorrCoefMulti(thresh = 0.5, sigmoid = TRUE, sample_weight = NULL)
```

Arguments

thresh	thresh
sigmoid	sigmoid
sample_weight	sample_weight

Value

None

max *Max*

Description

Max

Usage

```
## S3 method for class 'torch.Tensor'
max(a, ..., na.rm = FALSE)
```

Arguments

a	tensor
...	additional parameters
na.rm	remove NAs

Value

tensor

max.fastai.torch_core.TensorMask *Max*

Description

Max

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
max(a, ..., na.rm = FALSE)
```

Arguments

a	tensor
...	additional parameters
na.rm	remove NAs

Value

tensor

Examples

```
aa = tensor(1:10)
max(aa)
```

`MaxPool` *MaxPool*

Description

nn.MaxPool layer for ‘ndim‘

Usage

```
MaxPool(ks = 2, stride = NULL, padding = 0, ndim = 2, ceil_mode = FALSE)
```

Arguments

`ks`	kernel size
`stride`	the stride of the window. Default value is kernel_size
`padding`	implicit zero padding to be added on both sides
`ndim`	dimension number
`ceil_mode`	when True, will use ceil instead of floor to compute the output shape

Value

None

`maybe_unsqueeze` *Maybe_unsqueeze*

Description

Add empty dimension if it is a rank 1 tensor/array

Usage

```
maybe_unsqueeze(x)
```

Arguments

`x`	R array/matrix/tensor

Value

array

`mean.fastai.torch_core.TensorMask` *Mean of tensor*

Description

Mean of tensor

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
mean(x, ...)
```

Arguments

x	tensor
...	additional parameters to pass

Value

tensor

`mean.torch.Tensor` *Mean of tensor*

Description

Mean of tensor

Usage

```
## S3 method for class 'torch.Tensor'
mean(x, ...)
```

Arguments

x	tensor
...	additional parameters to pass

Value

tensor

medical *Medical module*

Description

Medical module

Usage

```
medical()
```

Value

None

MergeLayer *MergeLayer*

Description

Merge a shortcut with the result of the module by adding them or concatenating them if ‘dense=TRUE‘.

Usage

```
MergeLayer(dense = FALSE)
```

Arguments

dense	dense

Value

None

metrics *Metrics module*

Description

Metrics module

Usage

```
metrics()
```

Value

None

migrating_ignite *Ignite module*

Description

Ignite module

Usage

```
migrating_ignite()
```

Value

None

migrating_lightning *Lightning module*

Description

Lightning module

Usage

```
migrating_lightning()
```

Value

None

migrating_pytorch *Pytorch module*

Description

Pytorch module

Usage

```
migrating_pytorch()
```

Value

None

min *Min*

Description

Min

Usage

```
## S3 method for class 'torch.Tensor'
min(a, ..., na.rm = FALSE)
```

Arguments

a	tensor
...	additional parameters
na.rm	remove NAs

Value

tensor

min.fastai.torch_core.TensorMask *Min*

Description

Min

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
min(a, ..., na.rm = FALSE)
```

Arguments

a	tensor
...	additional parameters
na.rm	remove NAs

Value

tensor

mish *Mish*

Description

Mish

Usage

```
mish(x)
```

Arguments

x	tensor

Value

None

MishJitAutoFn *MishJitAutoFn*

Description

Records operation history and defines formulas for differentiating ops.

Usage

```
MishJitAutoFn(...)
```

Arguments

...	parameters to pass

Value

None

Mish_ *Class Mish*

Description

Class Mish

Usage

```
Mish_(...)
```

Arguments

...	parameters to pass

Value

None

Module *Module module*

Description

Module module

Usage

```
Module()
```

Value

None

Module_test *NN module*

Description

NN module

Usage

```
Module_test()
```

Value

None

momentum_step *Momentum_step*

Description

Step for SGD with momentum with ‘lr‘

Usage

```
momentum_step(p, lr, grad_avg, ...)
```

Arguments

p	p
lr	learning rate
grad_avg	grad average
...	additional arguments to pass

Value

None

most_confused *Most_confused*

Description

Sorted descending list of largest non-diagonal entries of confusion matrix, presented as actual, predicted, number of occurrences.

Usage

```
most_confused(interp, min_val = 1)
```

Arguments

interp	interpretation object
min_val	minimum value

Value

data frame

mse *MSE*

Description

Mean squared error between ‘inp‘ and ‘targ‘.

Usage

```
mse(inp, targ)
```

Arguments

inp	predictions
targ	targets

Value

None

Examples

```
## Not run:

model = dls %>% tabular_learner(layers=c(200,100,100,200),
metrics = list(mse(),rmse()) )


## End(Not run)
```

MSELossFlat *MSELossFlat*

Description

Flattens input and output, same as nn$MSELoss

Usage

```
MSELossFlat(...)
```

Arguments

...	parameters to pass

Value

Loss object

`msle` *MSLE*

Description

Mean squared logarithmic error between ‘inp‘ and ‘targ‘.

Usage

```
msle(inp, targ)
```

Arguments

`inp`	predictions
`targ`	targets

Value

None

`MultiCategorize` *MultiCategorize*

Description

Reversible transform of multi-category strings to ‘vocab‘ id

Usage

```
MultiCategorize(vocab = NULL, add_na = FALSE)
```

Arguments

`vocab`	vocabulary
`add_na`	add NA

Value

None

MultiCategoryBlock *MultiCategoryBlock*

Description

'TransformBlock' for multi-label categorical targets

Usage

```
MultiCategoryBlock(encoded = FALSE, vocab = NULL, add_na = FALSE)
```

Arguments

encoded	encoded or not
vocab	vocabulary
add_na	add NA

Value

Block object

---

multiplygit add -A && git commit -m 'staging all files'
Multiply

Description

Multiply

Usage

```
## S3 method for class 'torch.Tensor'
a * b
```

Arguments

a	tensor
b	tensor

Value

tensor

narrow *Modify tensor*

Description

Modify tensor

Usage

```
narrow(tensor, slice)
```

Arguments

tensor	torch tensor
slice	dimension

Value

tensor

Net *Net*

Description

Net model from Migrating_Pytorch

Usage

```
Net()
```

Value

model

Examples

```
## Not run:

Net()


## End(Not run)
```

nn *NN module*

Description

NN module

Usage

```
nn()
```

Value

None

nn_module *Fastai NN module*

Description

Fastai NN module

Usage

```
nn_module(model_fn)
```

Arguments

model_fn	pass custom model function

Value

None

NoiseColor *NoiseColor module*

Description

NoiseColor module

Usage

```
NoiseColor()
```

Value

None

NoneReduce *NoneReduce*

Description

A context manager to evaluate ‘loss_func‘ with none reduce.

Usage

```
NoneReduce(loss_func)
```

Arguments

loss_func	loss function

Value

None

noop *Noop*

Description

Noop

Usage

```
noop(...)
```

Arguments

...	parameters to pass

Value

None

Normalize *Normalize*

Description

Normalize the continuous variables.

Usage

```
Normalize(cat_names, cont_names)
```

Arguments

cat_names	cat_names
cont_names	cont_names

Value

None

NormalizeTS *NormalizeTS*

Description

Normalize the x variables.

Usage

```
NormalizeTS(enc = NULL, dec = NULL, split_idx = NULL, order = NULL)
```

Arguments

enc	encoder
dec	decoder
split_idx	split by index
order	order

Value

None

Normalize_from_stats *Normalize from stats*

Description

Normalize from stats

Usage

```
Normalize_from_stats(mean, std, dim = 1, ndim = 4, cuda = TRUE)
```

Arguments

mean	mean
std	standard deviation
dim	dimension
ndim	number of dimensions
cuda	cuda or not

Value

list

norm_apply_denorm *Norm_apply_denorm*

Description

Normalize ‘x‘ with ‘nrm‘, then apply ‘f‘, then denormalize

Usage

```
norm_apply_denorm(x, f, nrm)
```

Arguments

x	tensor
f	function
nrm	nrm

Value

None

not_equal_to *Not equal*

Description

Not equal

Usage

```
## S3 method for class 'torch.Tensor'
a != b
```

Arguments

a	tensor
b	tensor

Value

tensor

not_equal_to_mask_ *Not equal*

Description

Not equal

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a != b
```

Arguments

a	tensor
b	tensor

Value

tensor

not__mask *Logical_not*

Description

Logical_not

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
!x
```

Arguments

x	tensor

Value

tensor

Numericalize *Numericalize*

Description

Reversible transform of tokenized texts to numericalized ids

Usage

```
Numericalize(
  vocab = NULL,
  min_freq = 3,
  max_vocab = 60000,
  special_toks = NULL,
  pad_tok = NULL
)
```

Arguments

vocab	vocab
min_freq	min_freq
max_vocab	max_vocab
special_toks	special_toks
pad_tok	pad_tok

Value

None

`n_px` *N_px*

Description

int(x=0) -> integer

Usage

```
n_px(img)
```

Arguments

`img`	image

Value

None

`OldRandomCrop` *OldRandomCrop*

Description

Randomly crop an image to ‘size‘

Usage

```
OldRandomCrop(size, pad_mode = "zeros", ...)
```

Arguments

`size`	size
`pad_mode`	padding mode
`...`	additional arguments

Value

None

one_batch *One batch*

Description

One batch

Usage

```
one_batch(object, convert = FALSE)
```

Arguments

object	data loader
convert	to R matrix

Value

tensor

Examples

```
## Not run:

# get batch from data loader
batch = dls %>% one_batch()


## End(Not run)
```

OpenAudio *OpenAudio*

Description

Transform that creates AudioTensors from a list of files.

Usage

```
OpenAudio(items)
```

Arguments

items	vector, items

Value

None

`Optimizer` *Optimizer*

Description

Optimizer

Usage

```
Optimizer(...)
```

Arguments

`...`	parameters to pass

Value

None

`OptimWrapper` *OptimWrapper*

Description

OptimWrapper

Usage

```
OptimWrapper(...)
```

Arguments

`...`	parameters to pass

Value

None

optim_metric *Optim metric*

Description

Replace metric ‘f‘ with a version that optimizes argument ‘argname‘

Usage

```
optim_metric(f, argname, bounds, tol = 0.01, do_neg = TRUE, get_x = FALSE)
```

Arguments

f	f
argname	argname
bounds	bounds
tol	tol
do_neg	do_neg
get_x	get_x

Value

None

or_mask *Logical_or*

Description

Logical_or

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
x | y
```

Arguments

x	tensor
y	tensor

Value

tensor

os *Operating system*

Description

Operating system

Usage

```
os()
```

Value

vector

pad_conv_norm_relu *Pad_conv_norm_relu*

Description

Pad_conv_norm_relu

Usage

```
pad_conv_norm_relu(
  ch_in,
  ch_out,
  pad_mode,
  norm_layer,
  ks = 3,
  bias = TRUE,
  pad = 1,
  stride = 1,
  activ = TRUE,
  init = nn$init$kaiming_normal_,
  init_gain = 0.02
)
```

Arguments

ch_in	input
ch_out	output
pad_mode	padding mode
norm_layer	normalization layer

ks	kernel size
bias	bias
pad	padding
stride	stride
activ	activation
init	initializer
init_gain	init gain

Value

None

---

pad_input *Pad_input*

Description

Function that collect ‘samples‘ and adds padding

Usage

```
pad_input(
  samples,
  pad_idx = 1,
  pad_fields = 0,
  pad_first = FALSE,
  backwards = FALSE
)
```

Arguments

samples	samples
pad_idx	pad_idx
pad_fields	pad_fields
pad_first	pad_first
backwards	backwards

Value

None

pad_input_chunk *Pad_input_chunk*

Description

Pad ‘samples‘ by adding padding by chunks of size ‘seq_len‘

Usage

```
pad_input_chunk(samples, pad_idx = 1, pad_first = TRUE, seq_len = 72)
```

Arguments

samples	samples
pad_idx	pad_idx
pad_first	pad_first
seq_len	seq_len

Value

None

parallel *Parallel*

Description

Applies ‘func‘ in parallel to ‘items‘, using ‘n_workers‘

Usage

```
parallel(f, items, ...)
```

Arguments

f	file names
items	items
...	additional arguments

Value

None

parallel_tokenize *Parallel_tokenize*

Description

Calls optional ‘setup‘ on ‘tok‘ before launching ‘TokenizeWithRules‘ using ‘parallel_gen

Usage

```
parallel_tokenize(items, tok = NULL, rules = NULL, n_workers = 6)
```

Arguments

items	items
tok	tokenizer
rules	rules
n_workers	n_workers

Value

None

params *Params*

Description

Return all parameters of ‘m‘

Usage

```
params(m)
```

Arguments

m	parameters

Value

None

parent_label *Parent_label*

Description

Label ‘item‘ with the parent folder name.

Usage

```
parent_label(o)
```

Arguments

o	string, dir path

Value

vector

partial *Partial*

Description

partial(func, *args, **keywords) - new function with partial application

Usage

```
partial(...)
```

Arguments

...	additional arguments

Value

None

Examples

```
## Not run:

generator = basic_generator(out_size = 64, n_channels = 3, n_extra_layers = 1)
critic    = basic_critic(in_size = 64, n_channels = 3, n_extra_layers = 1,
                         act_cls = partial(nn$LeakyReLU, negative_slope = 0.2))


## End(Not run)
```

PartialLambda *Partial Lambda*

Description

Layer that applies ‘partial(func, ...)‘

Usage

```
PartialLambda(func)
```

Arguments

func	function

Value

None

---

pca *PCA*

Description

Compute PCA of ‘x‘ with ‘k‘ dimensions.

Usage

```
pca(object, k = 3, convert = TRUE)
```

Arguments

object	an object to apply PCA
k	number of dimensions
convert	to R matrix

Value

tensor

`PearsonCorrCoef` *PearsonCorrCoef*

Description

Pearson correlation coefficient for regression problem

Usage

```
PearsonCorrCoef(
  dim_argmax = NULL,
  activation = "no",
  thresh = NULL,
  to_np = FALSE,
  invert_arg = FALSE,
  flatten = TRUE
)
```

Arguments

`dim_argmax`	dim_argmax
`activation`	activation
`thresh`	thresh
`to_np`	to_np
`invert_arg`	invert_arg
`flatten`	flatten

Value

None

Perplexity *Perplexity*

Description

Perplexity

Usage

```
Perplexity(...)
```

Arguments

...	parameters to pass

Value

None

Pipeline *Pipeline*

Description

A pipeline of composed (for encode/decode) transforms, setup with types

Usage

```
Pipeline(funcs = NULL, split_idx = NULL)
```

Arguments

funcs	functions
split_idx	split by index

Value

None

PixelShuffle_ICNR *PixelShuffle_ICNR*

Description

Upsample by ‘scale‘ from ‘ni‘ filters to ‘nf‘ (default ‘ni‘), using ‘nn.PixelShuffle‘.

Usage

```
PixelShuffle_ICNR(
  ni,
  nf = NULL,
  scale = 2,
  blur = FALSE,
  norm_type = 3,
  act_cls = nn$ReLU
)
```

Arguments

ni	input shape
nf	number of features / outputs
scale	scale
blur	blur
norm_type	normalziation type
act_cls	activation

Value

None

plot *Plot dicom*

Description

Plot dicom

Usage

```
plot(x, y, ..., dpi = 100)
```

Arguments

x	model
y	y axis
...	parameters to pass
dpi	dots per inch

Value

None

---

plot_bs_find *Plot_bs_find*

Description

Plot_bs_find

Usage

```
plot_bs_find(object, ..., dpi = 250)
```

Arguments

object	model
...	additional arguments
dpi	dots per inch

Value

None

---

plot_confusion_matrix *Plot_confusion_matrix*

Description

Plot the confusion matrix, with ‘title‘ and using ‘cmap‘.

Usage

```
plot_confusion_matrix(
  interp,
  normalize = FALSE,
  title = "Confusion matrix",
  cmap = "Blues",
  norm_dec = 2,
  plot_txt = TRUE,
  figsize = c(19.2, 10.8),
  ...,
  dpi = 250
)
```

Arguments

interp	interpretation object
normalize	normalize
title	title
cmap	color map
norm_dec	norm dec
plot_txt	plot text
figsize	plot size
...	additional parameters to pass
dpi	dots per inch

Value

None

Examples

```
## Not run:

interp = ClassificationInterpretation_from_learner(model)
interp %>% plot_confusion_matrix(dpi = 90,figsize = c(6,6))


## End(Not run)
```

plot_loss *Plot_loss*

Description

Plot the losses from ‘skip_start‘ and onward

Usage

```
plot_loss(object, skip_start = 5, with_valid = TRUE, dpi = 200)
```

Arguments

object	model
skip_start	n points to skip the start
with_valid	with validation
dpi	dots per inch

Value

None

plot_lr_find *Plot_lr_find*

Description

Plot the result of an LR Finder test (won’t work if you didn’t do ‘lr_find(learn)‘ before)

Usage

```
plot_lr_find(object, skip_end = 5, dpi = 250)
```

Arguments

object	model
skip_end	n points to skip the end
dpi	dots per inch

Value

None

plot_top_losses *Plot_top_losses*

Description

Plot_top_losses

Usage

```
plot_top_losses(
  interp,
  k,
  largest = TRUE,
  figsize = c(19.2, 10.8),
  ...,
  dpi = NULL
)
```

Arguments

interp	interpretation object
k	number of images
largest	largest
figsize	plot size
...	additional parameters to pass
dpi	dots per inch

Value

None

Examples

```
## Not run:

# get interperetation from learn object, the model.
interp = ClassificationInterpretation_from_learner(learn)
interp %>% plot_top_losses(k = 9, figsize = c(15,11))


## End(Not run)
```

PointBlock *PointBlock*

Description

A ‘TransformBlock‘ for points in an image

Usage

```
PointBlock()
```

Value

None

PointScaler *PointScaler*

Description

Scale a tensor representing points

Usage

```
PointScaler(do_scale = TRUE, y_first = FALSE)
```

Arguments

do_scale	do scale
y_first	y first

Value

None

PooledSelfAttention2d *PooledSelfAttention2d*

Description

Pooled self attention layer for 2d.

Usage

```
PooledSelfAttention2d(n_channels)
```

Arguments

n_channels	number of channels

Value

None

PoolFlatten *PoolFlatten*

Description

Combine ‘nn.AdaptiveAvgPool2d‘ and ‘Flatten‘.

Usage

```
PoolFlatten(pool_type = "Avg")
```

Arguments

pool_type	pooling type

Value

None

PoolingLinearClassifier *PoolingLinearClassifier*

Description

Create a linear classifier with pooling

Usage

```
PoolingLinearClassifier(dims, ps, bptt, y_range = NULL)
```

Arguments

dims	dims
ps	ps
bptt	bptt
y_range	y_range

Value

None

pow *Pow*

Description

Pow

Usage

```
## S3 method for class 'torch.Tensor'
a ^ b
```

Arguments

a	tensor
b	tensor

Value

tensor

Precision *Precision*

Description

Precision for single-label classification problems

Usage

```
Precision(
  axis = -1,
  labels = NULL,
  pos_label = 1,
  average = "binary",
  sample_weight = NULL
)
```

Arguments

axis	axis
labels	labels
pos_label	pos_label
average	average
sample_weight	sample_weight

Value

None

PrecisionMulti *PrecisionMulti*

Description

Precision for multi-label classification problems

Usage

```
PrecisionMulti(
  thresh = 0.5,
  sigmoid = TRUE,
  labels = NULL,
  pos_label = 1,
  average = "macro",
  sample_weight = NULL
)
```

Arguments

thresh	thresh
sigmoid	sigmoid
labels	labels
pos_label	pos_label
average	average
sample_weight	sample_weight

Value

None

---

predict.fastai.learner.Learner
Predict

Description

Prediction on 'item', fully decoded, loss function decoded and probabilities

Usage

```
## S3 method for class 'fastai.learner.Learner'
predict(object, row, ...)
```

Arguments

object	the model
row	row
...	additional arguments to pass

Value

data frame

predict.fastai.tabular.learner.TabularLearner *Predict*

Description

Prediction on 'item', fully decoded, loss function decoded and probabilities

Usage

```
## S3 method for class 'fastai.tabular.learner.TabularLearner'
predict(object, row, ...)
```

Arguments

object	the model
row	row
...	additional arguments to pass

Value

data frame

preplexity *Perplexity*

Description

Perplexity (exponential of cross-entropy loss) for Language Models

Usage

```
preplexity(...)
```

Arguments

...	parameters to pass

Value

None

PreprocessAudio *Preprocess Audio*

Description

Creates an audio tensor and run the basic preprocessing transforms on it.

Usage

```
PreprocessAudio(sample_rate = 16000, force_mono = TRUE, crop_signal_to = NULL)
```

Arguments

sample_rate	sample rate
force_mono	force mono or not
crop_signal_to	int, crop signal

Details

Used while preprocessing the audios, this is not a 'Transform'.

Value

None

preprocess_audio_folder *Preprocess audio folder*

Description

Preprocess audio files in 'path' in parallel using 'n_workers'

Usage

```
preprocess_audio_folder(
  path,
  folders = NULL,
  output_dir = NULL,
  sample_rate = 16000,
  force_mono = TRUE,
  crop_signal_to = NULL
)
```

Arguments

path	directory, path
folders	folders
output_dir	output directory
sample_rate	sample rate
force_mono	force mono or not
crop_signal_to	int, crop signal

Value

None

`print.fastai.learner.Learner`
Print model

Description

Print model

Usage

```
## S3 method for class 'fastai.learner.Learner'
print(x, ...)
```

Arguments

x	object
...	additional parameters to pass

Value

None

print.fastai.tabular.learner.TabularLearner
Print tabular model

Description

Print tabular model

Usage

```
## S3 method for class 'fastai.tabular.learner.TabularLearner'
print(x, ...)
```

Arguments

x	model
...	additional parameters to pass

Value

None

print.pydicom.dataset.FileDataset
Dicom

Description

prints dicom file

Usage

```
## S3 method for class 'pydicom.dataset.FileDataset'
print(x, ...)
```

Arguments

x	dicom file
...	additional parameters to pass

Value

None

python_path *Python path*

Description

Python path

Usage

```
python_path()
```

Value

None

QHAdam *QHAdam*

Description

QHAdam

Usage

```
QHAdam(...)
```

Arguments

...	parameters to pass

Value

None

qhadam_step *Qhadam_step*

Description

Qhadam_step

Usage

```
qhadam_step(p, lr, mom, sqr_mom, sqr_avg, nu_1, nu_2, step, grad_avg, eps, ...)
```

Arguments

p	p
lr	learning rate
mom	momentum
sqr_mom	sqr momentum
sqr_avg	sqr average
nu_1	nu_1
nu_2	nu_2
step	step
grad_avg	gradient average
eps	epsilon
...	additional arguments to pass

Value

None

QRNN *QRNN*

Description

Apply a multiple layer Quasi-Recurrent Neural Network (QRNN) to an input sequence.

Usage

```
QRNN(
  input_size,
  hidden_size,
  n_layers = 1,
  batch_first = TRUE,
  dropout = 0,
  bidirectional = FALSE,
  save_prev_x = FALSE,
  zoneout = 0,
  window = NULL,
  output_gate = TRUE
)
```

Arguments

input_size	input_size
hidden_size	hidden_size
n_layers	n_layers
batch_first	batch_first
dropout	dropout
bidirectional	bidirectional
save_prev_x	save_prev_x
zoneout	zoneout
window	window
output_gate	output_gate

Value

None

QRNNLayer *QRNNLayer*

Description

Apply a single layer Quasi-Recurrent Neural Network (QRNN) to an input sequence.

Usage

```
QRNNLayer(
  input_size,
  hidden_size = NULL,
  save_prev_x = FALSE,
  zoneout = 0,
  window = 1,
  output_gate = TRUE,
  batch_first = TRUE,
  backward = FALSE
)
```

Arguments

input_size	input_size
hidden_size	hidden_size
save_prev_x	save_prev_x
zoneout	zoneout
window	window
output_gate	output_gate
batch_first	batch_first
backward	backward

Value

None

R2Score *R2Score*

Description

R2 score between predictions and targets

Usage

```
R2Score(sample_weight = NULL)
```

Arguments

sample_weight	sample_weight

Value

None

RAdam *RAdam*

Description

RAdam

Usage

```
RAdam(...)
```

Arguments

...	parameters to pass

Value

None

radam_step *Radam_step*

Description

Step for RAdam with ‘lr‘ on ‘p‘

Usage

```
radam_step(p, lr, mom, step, sqr_mom, grad_avg, sqr_avg, eps, beta, ...)
```

Arguments

p	p
lr	learning rate
mom	momentum
step	step
sqr_mom	sqr momentum
grad_avg	grad average
sqr_avg	sqr average
eps	epsilon
beta	beta
...	additional arguments to pass

Value

None

RandomCrop *RandomCrop*

Description

Randomly crop an image to 'size'

Usage

```
RandomCrop(size, ...)
```

Arguments

size	size
...	additional arguments

Value

None

RandomErasing *RandomErasing*

Description

Randomly selects a rectangle region in an image and randomizes its pixels.

Usage

```
RandomErasing(p = 0.5, sl = 0, sh = 0.3, min_aspect = 0.3, max_count = 1)
```

Arguments

p	probability
sl	sl
sh	sh
min_aspect	minimum aspect
max_count	maximum count

Value

None

RandomResizedCrop *RandomResizedCrop*

Description

Picks a random scaled crop of an image and resize it to 'size'

Usage

```
RandomResizedCrop(
  size,
  min_scale = 0.08,
  ratio = list(0.75, 1.33333333333333),
  resamples = list(2, 0),
  val_xtra = 0.14
)
```

Arguments

size	size
min_scale	minimum scale
ratio	ratio
resamples	resamples
val_xtra	validation xtra

Value

None

RandomResizedCropGPU *RandomResizedCropGPU*

Description

Picks a random scaled crop of an image and resize it to 'size'

Usage

```
RandomResizedCropGPU(
  size,
  min_scale = 0.08,
  ratio = list(0.75, 1.33333333333333),
  mode = "bilinear",
  valid_scale = 1
)
```

Arguments

size	size
min_scale	minimum scale
ratio	ratio
mode	mode
valid_scale	validation scale

Value

None

RandomSplitter *RandomSplitter*

Description

Create function that splits ‘items‘ between train/val with ‘valid_pct‘ randomly.

Usage

```
RandomSplitter(valid_pct = 0.2, seed = NULL)
```

Arguments

valid_pct	validation percenatge split
seed	random seed

Value

None

RandPair *RandPair*

Description

Returns a random image from domain B, resulting in a random pair of images from domain A and B.

Usage

```
RandPair(itemsB)
```

Arguments

itemsB	a random image from domain B

Value

None

RandTransform *RandTransform*

Description

A transform that before_call its state at each '__call__'

Usage

```
RandTransform(p = 1, nm = NULL, before_call = NULL, ...)
```

Arguments

p	probability
nm	nm
before_call	before call
...	additional arguments to pass

Value

None

---

ranger *Ranger*

Description

Convenience method for 'Lookahead' with 'RAdam'

Usage

```
ranger(
  p,
  lr,
  mom = 0.95,
  wd = 0.01,
  eps = 1e-06,
  sqr_mom = 0.99,
  beta = 0,
  decouple_wd = TRUE
)
```

Arguments

p	p
lr	learning rate
mom	momentum
wd	weight decay
eps	epsilon
sqr_mom	sqr momentum
beta	beta
decouple_wd	decouple weight decay

Value

None

`RatioResize` *RatioResize*

Description

Resizes the biggest dimension of an image to ‘max_sz‘ maintaining the aspect ratio

Usage

```
RatioResize(max_sz, resamples = list(2, 0), ...)
```

Arguments

max_sz	maximum sz
resamples	resamples
...	additional arguments

Value

None

ReadTSBatch *ReadTSBatch*

Description

A transform that always take lists as items

Usage

```
ReadTSBatch(to)
```

Arguments

to	output from TSDataTable function

Value

None

Recall *Recall*

Description

Recall for single-label classification problems

Usage

```
Recall(
  axis = -1,
  labels = NULL,
  pos_label = 1,
  average = "binary",
  sample_weight = NULL
)
```

Arguments

axis	axis
labels	labels
pos_label	pos_label
average	average
sample_weight	sample_weight

Value

None

RecallMulti *RecallMulti*

Description

Recall for multi-label classification problems

Usage

```
RecallMulti(
  thresh = 0.5,
  sigmoid = TRUE,
  labels = NULL,
  pos_label = 1,
  average = "macro",
  sample_weight = NULL
)
```

Arguments

thresh	thresh
sigmoid	sigmoid
labels	labels
pos_label	pos_label
average	average
sample_weight	sample_weight

Value

None

ReduceLROnPlateau *ReduceLROnPlateau*

Description

ReduceLROnPlateau

Usage

```
ReduceLROnPlateau(...)
```

Arguments

...	parameters to pass

Value

None

Examples

```
## Not run:

URLs_MNIST_SAMPLE()
# transformations
tfms = aug_transforms(do_flip = FALSE)
path = 'mnist_sample'
bs = 20

#load into memory
data = ImageDataLoaders_from_folder(path, batch_tfms = tfms, size = 26, bs = bs)


learn = cnn_learner(data, resnet18(), metrics = accuracy, path = getwd())

learn %>% fit_one_cycle(10, 1e-2, cbs = ReduceLROnPlateau(monitor='valid_loss', patience = 1))


## End(Not run)
```

`RegressionBlock` *RegressionBlock*

Description

'TransformBlock' for float targets

Usage

```
RegressionBlock(n_out = NULL)
```

Arguments

`n_out`	number of out features

Value

Block object

RemoveSilence *Remove Silence*

Description

Split signal at points of silence greater than 2*pad_ms

Usage

```
RemoveSilence(
  remove_type = RemoveType()$Trim$value,
  threshold = 20,
  pad_ms = 20
)
```

Arguments

remove_type	remove type from RemoveType module
threshold	threshold point
pad_ms	pad milliseconds

Value

None

RemoveType *RemoveType module*

Description

RemoveType module

Usage

```
RemoveType()
```

Value

None

`replace_all_caps` *Replace_all_caps*

Description

Replace tokens in ALL CAPS by their lower version and add ‘TK_UP‘ before.

Usage

```
replace_all_caps(t)
```

Arguments

t	text

Value

string

`replace_maj` *Replace_maj*

Description

Replace tokens in ALL CAPS by their lower version and add ‘TK_UP‘ before.

Usage

```
replace_maj(t)
```

Arguments

t	text

Value

string

replace_rep *Replace_rep*

Description

Replace repetitions at the character level: cccc – TK_REP 4 c

Usage

```
replace_rep(t)
```

Arguments

t	text

Value

string

replace_wrep *Replace_wrep*

Description

Replace word repetitions: word word word word – TK_WREP 4 word

Usage

```
replace_wrep(t)
```

Arguments

t	text

Value

string

Resample *Resample*

Description

Resample using faster polyphase technique and avoiding FFT computation

Usage

```
Resample(sr_new)
```

Arguments

sr_new	input

Value

None

ResBlock *ResBlock*

Description

Resnet block from ‘ni‘ to ‘nh‘ with ‘stride‘

Usage

```
ResBlock(
  expansion,
  ni,
  nf,
  stride = 1,
  groups = 1,
  reduction = NULL,
  nh1 = NULL,
  nh2 = NULL,
  dw = FALSE,
  g2 = 1,
  sa = FALSE,
  sym = FALSE,
  norm_type = 1,
  act_cls = nn$ReLU,
  ndim = 2,
  ks = 3,
  pool = AvgPool(),
```

```
  pool_first = TRUE,
  padding = NULL,
  bias = NULL,
  bn_1st = TRUE,
  transpose = FALSE,
  init = "auto",
  xtra = NULL,
  bias_std = 0.01,
  dilation = 1,
  padding_mode = "zeros"
)
```

Arguments

expansion	decoder
ni	number of linear inputs
nf	number of features
stride	stride number
groups	groups number
reduction	reduction
nh1	out channels 1
nh2	out channels 2
dw	dw paramer
g2	g2 block
sa	sa parameter
sym	symmetric
norm_type	normalization type
act_cls	activation
ndim	dimension number
ks	kernel size
pool	pooling type, Average, Max
pool_first	pooling first
padding	padding
bias	bias
bn_1st	batch normalization 1st
transpose	transpose
init	initializer
xtra	xtra
bias_std	bias standard deviation
dilation	dilation number
padding_mode	padding mode

Value

Block object

`reshape` *Reshape*

Description

resize x to (w,h)

Usage

```
reshape(x, h, w, resample = 0)
```

Arguments

`x`	tensor
`h`	height
`w`	width
`resample`	resample value

Value

None

`Resize` *Resize*

Description

A transform that before_call its state at each '__call__'

Usage

```
Resize(size, method = "crop", pad_mode = "reflection", resamples = list(2, 0))
```

Arguments

`size`	size of image
`method`	method
`pad_mode`	reflection, zeros, border as string parameter
`resamples`	list of integers

Value

None

ResizeBatch *ResizeBatch*

Description

Reshape x to size, keeping batch dim the same size

Usage

```
ResizeBatch(...)
```

Arguments

...	parameters to pass

Value

None

ResizeSignal *Resize Signal*

Description

Crops signal to be length specified in ms by duration, padding if needed

Usage

```
ResizeSignal(duration, pad_mode = AudioPadType()$Zeros)
```

Arguments

duration	int, duration
pad_mode	padding mode

Value

None

resize_max *Resize_max*

Description

'resize' 'x' to 'max_px', or 'max_h', or 'max_w'

Usage

```
resize_max(img, resample = 0, max_px = NULL, max_h = NULL, max_w = NULL)
```

Arguments

img	image
resample	resample value
max_px	max px
max_h	max height
max_w	max width

Value

None

ResNet *ResNet*

Description

Base class for all neural network modules.

Usage

```
ResNet(
  block,
  layers,
  num_classes = 1000,
  zero_init_residual = FALSE,
  groups = 1,
  width_per_group = 64,
  replace_stride_with_dilation = NULL,
  norm_layer = NULL
)
```

Arguments

block	the blocks that need to passed to ResNet
layers	the layers to pass to ResNet
num_classes	the number of classes
zero_init_residual	logical, initializer
groups	the groups
width_per_group	the width per group
replace_stride_with_dilation	logical, replace stride with dilation
norm_layer	norm_layer

---

resnet101 *Resnet101*

Description

ResNet-101 model from

Usage

```
resnet101(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Deep Residual Learning for Image Recognition" <https://arxiv.org/pdf/1512.03385.pdf>

Value

model

resnet152 *Resnet152*

Description

Resnet152

Usage

```
resnet152(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Deep Residual Learning for Image Recognition" <https://arxiv.org/pdf/1512.03385.pdf>

Value

model

resnet18 *Resnet18*

Description

Resnet18

Usage

```
resnet18(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Deep Residual Learning for Image Recognition" <https://arxiv.org/pdf/1512.03385.pdf>

Value

model

resnet34 *Resnet34*

Description

ResNet-34 model from

Usage

```
resnet34(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Deep Residual Learning for Image Recognition" <https://arxiv.org/pdf/1512.03385.pdf>

Value

model

resnet50 *Resnet50*

Description

Resnet50

Usage

```
resnet50(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Deep Residual Learning for Image Recognition" <https://arxiv.org/pdf/1512.03385.pdf>

Value

model

ResnetBlock *ResnetBlock*

Description

nn$Module for the ResNet Block

Usage

```
ResnetBlock(
  dim,
  pad_mode = "reflection",
  norm_layer = NULL,
  dropout = 0,
  bias = TRUE
)
```

Arguments

dim	dimension
pad_mode	padding mode
norm_layer	normalization layer
dropout	dropout rate
bias	bias or not

Value

None

resnet_generator *Resnet_generator*

Description

Resnet_generator

Usage

```
resnet_generator(
  ch_in,
  ch_out,
  n_ftrs = 64,
  norm_layer = NULL,
  dropout = 0,
  n_blocks = 9,
  pad_mode = "reflection"
)
```

Arguments

ch_in	input
ch_out	output
n_ftrs	filter
norm_layer	normalziation layer
dropout	dropout rate
n_blocks	number of blocks
pad_mode	paddoing mode

Value

None

---

res_block_1d *Res_block_1d*

Description

Resnet block as described in the paper.

Usage

```
res_block_1d(nf, ks = c(5, 3))
```

Arguments

nf	number of features
ks	kernel size

Value

block

RetinaNet *RetinaNet*

Description

Implements RetinaNet from https://arxiv.org/abs/1708.02002

Usage

```
RetinaNet(...)
```

Arguments

...	arguments to pass

Value

model

Examples

```
## Not run:

encoder = create_body(resnet34(), pretrained = TRUE)
arch = RetinaNet(encoder, get_c(dls), final_bias=-4)


## End(Not run)
```

RetinaNetFocalLoss *RetinaNetFocalLoss*

Description

Base class for all neural network modules.

Usage

```
RetinaNetFocalLoss(...)
```

Arguments

...	parameters to pass

Details

Your models should also subclass this class. Modules can also contain other Modules, allowing to nest them in a tree structure. You can assign the submodules as regular attributes:: import torch.nn as nn import torch.nn.functional as F class Model(nn.Module): def __init__(self): super(Model, self).__init__() self.conv1 = nn.Conv2d(1, 20, 5) self.conv2 = nn.Conv2d(20, 20, 5) def forward(self, x): x = F.relu(self.conv1(x)) return F.relu(self.conv2(x)) Submodules assigned in this way will be registered, and will have their parameters converted too when you call :meth:‘to‘, etc.

Value

None

---

retinanet_ *Retinanet module*

Description

Retinanet module

Usage

```
retinanet_()
```

Value

None

---

reverse_text *Reverse_text*

Description

Reverse_text

Usage

```
reverse_text(x)
```

Arguments

x	text

Value

string

rgb2hsv *Rgb2hsv*

Description

Converts a RGB image to an HSV image.

Usage

```
rgb2hsv(img)
```

Arguments

img	image object

Details

Note: Will not work on logit space images.

Value

None

rmse *RMSE*

Description

Root mean squared error

Usage

```
rmse(preds, targs)
```

Arguments

preds	predictions
targs	targets

Value

None

Examples

```
## Not run:

model = dls %>% tabular_learner(layers=c(200,100,100,200),
metrics = list(mse(),rmse()) )


## End(Not run)
```

`RMSProp` *RMSProp*

Description

RMSProp

Usage

```
RMSProp(...)
```

Arguments

`...`	parameters to pass

Value

None

`rms_prop_step` *Rms_prop_step*

Description

Step for SGD with momentum with ‘lr‘

Usage

```
rms_prop_step(p, lr, sqr_avg, eps, grad_avg = NULL, ...)
```

Arguments

p	p
lr	learning rate
sqr_avg	sqr average
eps	epsilon
grad_avg	grad average
...	additional arguments to pass

Value

None

---

rm_useless_spaces *Rm_useless_spaces*

Description

Remove multiple spaces

Usage

```
rm_useless_spaces(t)
```

Arguments

t	text

Value

string

Examples

```
## Not run:

rm_useless_spaces('hello,   Sir!')


## End(Not run)
```

RNNDropout *RNNDropout*

Description

Dropout with probability ‘p‘ that is consistent on the seq_len dimension.

Usage

```
RNNDropout(p = 0.5)
```

Arguments

p	p

Value

None

RocAuc *RocAuc*

Description

Area Under the Receiver Operating Characteristic Curve for single-label multiclass classification problems

Usage

```
RocAuc(
  axis = -1,
  average = "macro",
  sample_weight = NULL,
  max_fpr = NULL,
  multi_class = "ovr"
)
```

Arguments

axis	axis
average	average
sample_weight	sample_weight
max_fpr	max_fpr
multi_class	multi_class

Value

None

RocAucBinary *RocAucBinary*

Description

Area Under the Receiver Operating Characteristic Curve for single-label binary classification problems

Usage

```
RocAucBinary(
  axis = -1,
  average = "macro",
  sample_weight = NULL,
  max_fpr = NULL,
  multi_class = "raise"
)
```

Arguments

axis	axis
average	average
sample_weight	sample_weight
max_fpr	max_fpr
multi_class	multi_class

Value

None

Examples

```
## Not run:

model = dls %>% tabular_learner(layers=c(200,100,100,200),
config = tabular_config(embed_p = 0.3, use_bn = FALSE),
metrics = list(accuracy, RocAucBinary(),
               Precision(), Recall(),
               F1Score()))


## End(Not run)
```

RocAucMulti *RocAucMulti*

Description

Area Under the Receiver Operating Characteristic Curve for multi-label binary classification problems

Usage

```
RocAucMulti(
  sigmoid = TRUE,
  average = "macro",
  sample_weight = NULL,
  max_fpr = NULL
)
```

Arguments

sigmoid	sigmoid
average	average
sample_weight	sample_weight
max_fpr	max_fpr

Value

None

Rotate *Rotate*

Description

Apply a random rotation of at most ‘max_deg‘ with probability ‘p‘ to a batch of images

Usage

```
Rotate(
  max_deg = 10,
  p = 0.5,
  draw = NULL,
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  align_corners = TRUE,
  batch = FALSE
)
```

Arguments

max_deg	maximum degrees
p	probability
draw	draw
size	size of image
mode	mode
pad_mode	reflection, zeros, border as string parameter
align_corners	align corners or not
batch	batch or not

Value

None

---

rotate_mat *Rotate_mat*

Description

Return a random rotation matrix with ‘max_deg‘ and ‘p‘

Usage

```
rotate_mat(x, max_deg = 10, p = 0.5, draw = NULL, batch = FALSE)
```

Arguments

x	tensor
max_deg	max_deg
p	probability
draw	draw
batch	batch

Value

None

round *Round*

Description

Round

Usage

```
## S3 method for class 'torch.Tensor'
round(x, digits = 0)
```

Arguments

x	tensor
digits	decimal

Value

tensor

round.fastai.torch_core.TensorMask *Round*

Description

Round

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
round(x, digits = 0)
```

Arguments

x	tensor
digits	decimal

Value

tensor

Saturation *Saturation*

Description

Apply change in saturation of ‘max_lighting‘ to batch of images with probability ‘p‘.

Usage

```
Saturation(max_lighting = 0.2, p = 0.75, draw = NULL, batch = FALSE)
```

Arguments

max_lighting	maximum lighting
p	probability
draw	draw
batch	batch

Value

None

SaveModelCallback *SaveModelCallback*

Description

SaveModelCallback

Usage

```
SaveModelCallback(...)
```

Arguments

...	parameters to pass

Value

None

SEBlock *SEBlock*

Description

SEBlock

Usage

```
SEBlock(expansion, ni, nf, groups = 1, reduction = 16, stride = 1)
```

Arguments

expansion	decoder
ni	number of inputs
nf	number of features
groups	number of groups
reduction	number of reduction
stride	number of strides

Value

Block object

SegmentationDataLoaders_from_label_func *SegmentationDataLoaders_from_label_func*

Description

Create from list of ‘fnames‘ in ‘path‘s with ‘label_func‘.

Usage

```
SegmentationDataLoaders_from_label_func(
  path,
  fnames,
  label_func,
  valid_pct = 0.2,
  seed = NULL,
  codes = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  bs = 64,
```

```
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL
)
```

Arguments

path	path
fnames	file names
label_func	label function
valid_pct	validation percentage
seed	seed
codes	codes
item_tfms	item transformations
batch_tfms	batch transformations
bs	batch size
val_bs	validation batch size
shuffle_train	shuffle train
device	device name

Value

None

`SelfAttention` *SelfAttention*

Description

Self attention layer for ‘n_channels‘.

Usage

```
SelfAttention(n_channels)
```

Arguments

n_channels	number of channels

Value

None

SEModule *SEModule*

Description

SEModule

Usage

```
SEModule(ch, reduction, act_cls = nn$ReLU)
```

Arguments

ch	ch
reduction	reduction
act_cls	activation

Value

None

SentenceEncoder *SentenceEncoder*

Description

Create an encoder over ‘module‘ that can process a full sentence.

Usage

```
SentenceEncoder(bptt, module, pad_idx = 1, max_len = NULL)
```

Arguments

bptt	bptt
module	module
pad_idx	pad_idx
max_len	max_len

Value

None

SentencePieceTokenizer *SentencePieceTokenizer*

Description

SentencePiece tokenizer for ‘lang‘

Usage

```
SentencePieceTokenizer(
  lang = "en",
  special_toks = NULL,
  sp_model = NULL,
  vocab_sz = NULL,
  max_vocab_sz = 30000L,
  model_type = "unigram",
  char_coverage = NULL,
  cache_dir = "tmp"
)
```

Arguments

lang	lang
special_toks	special_toks
sp_model	sp_model
vocab_sz	vocab_sz
max_vocab_sz	max_vocab_sz
model_type	model_type
char_coverage	char_coverage
cache_dir	cache_dir

Value

None

SeparableBlock *SeparableBlock*

Description

SeparableBlock

Usage

```
SeparableBlock(expansion, ni, nf, reduction = 16, stride = 1, base_width = 4)
```

Arguments

expansion	decoder
ni	number of inputs
nf	number of features
reduction	number of reduction
stride	number of stride
base_width	base width

Value

Block object

sequential *Sequential*

Description

Sequential

Usage

```
sequential(...)
```

Arguments

...	parameters to pass

Value

None

SequentialEx *SequentialEx*

Description

SequentialEx

Usage

```
SequentialEx(...)
```

Arguments

...	parameters to pass

Value

None

SequentialRNN *Sequential RNN*

Description

Sequential RNN

Usage

```
SequentialRNN(...)
```

Arguments

...	parameters to pass

Value

layer

SEResNeXtBlock *SEResNeXtBlock*

Description

SEResNeXtBlock

Usage

```
SEResNeXtBlock(
  expansion,
  ni,
  nf,
  groups = 32,
  reduction = 16,
  stride = 1,
  base_width = 4
)
```

Arguments

expansion	decoder
ni	number of linear inputs
nf	number of features
groups	groups number
reduction	reduction number
stride	stride number
base_width	int, base width

Value

Block object

setup_aug_tfms *Setup_aug_tfms*

Description

Go through ‘tfms‘ and combines together affine/coord or lighting transforms

Usage

```
setup_aug_tfms(tfms)
```

Arguments

tfms	transformations

Value

None

---

set_freeze_model *Set freeze model*

Description

Set freeze model

Usage

```
set_freeze_model(m, rg)
```

Arguments

m	parameters
rg	rg

Value

None

---

set_item_pg *Set_item_pg*

Description

Set_item_pg

Usage

```
set_item_pg(pg, k, v)
```

Arguments

pg	pg
k	k
v	v

Value

None

SGD *SGD*

Description

SGD

Usage

```
SGD(...)
```

Arguments

...	parameters to pass

Value

None

sgd_step *Sgd_step*

Description

Sgd_step

Usage

```
sgd_step(p, lr, ...)
```

Arguments

p	p
lr	learning rate
...	additional arguments to pass

Value

None

Examples

```
## Not run:

tst_param = function(val, grad = NULL) {
  "Create a tensor with `val` and a gradient of `grad` for testing"
  res = tensor(val) %>% float()

  if(is.null(grad)) {
    grad = tensor(val / 10)
  } else {
    grad = tensor(grad)
  }

  res$grad = grad %>% float()
  res
}
p = tst_param(1., 0.1)
sgd_step(p, 1.)


## End(Not run)
```

SGRoll *SGRoll*

Description

Shifts spectrogram along x-axis wrapping around to other side

Usage

```
SGRoll(max_shift_pct = 0.5, direction = 0)
```

Arguments

max_shift_pct	maximum shift percentage
direction	direction

Value

None

shap *Shap module*

Description

Shap module

Usage

```
shap()
```

Value

None

shape *Shape*

Description

Shape

Usage

```
shape(img)
```

Arguments

img	image

Value

None

ShapInterpretation *ShapInterpretation*

Description

Base interpereter to use the 'SHAP' interpretation library

Usage

```
ShapInterpretation(
  learn,
  test_data = NULL,
  link = "identity",
  l1_reg = "auto",
  n_samples = 128
)
```

Arguments

learn	learner/model
test_data	should be either a Pandas dataframe or a TabularDataLoader. If not, 100 random rows of the training data will be used instead.
link	link can either be "identity" or "logit". A generalized linear model link to connect the feature importance values to the model output. Since the feature importance values, phi, sum up to the model output, it often makes sense to connect them to the ouput with a link function where link(outout) = sum(phi). If the model output is a probability then the LogitLink link function makes the feature importance values have log-odds units.
l1_reg	can be an integer value representing the number of features, "auto", "aic", "bic", or a float value. The l1 regularization to use for feature selection (the estimation procedure is based on a debiased lasso). The auto option currently uses "aic" when less that 20 space is enumerated, otherwise it uses no regularization.
n_samples	can either be "auto" or an integer value. This is the number of times to re-evaluate the model when explaining each predictions. More samples leads to lower variance estimations of the SHAP values

Value

None

Shortcut *Shortcut*

Description

Merge a shortcut with the result of the module by adding them. Adds Conv, BN and ReLU

Usage

```
Shortcut(ni, nf, act_fn = nn$ReLU(inplace = TRUE))
```

Arguments

ni	number of input channels
nf	number of features
act_fn	activation

Value

None

ShortEpochCallback *ShortEpochCallback*

Description

ShortEpochCallback

Usage

```
ShortEpochCallback(...)
```

Arguments

...	parameters to pass

Value

None

show *Show*

Description

Adds functionality to view dicom images where each file may have more than 1 frame

Usage

```
show(img, frames = 1, scale = TRUE, ...)
```

Arguments

img	image object
frames	number of frames
scale	scale
...	additional arguments

Value

None

ShowCycleGANImgsCallback *ShowCycleGANImgsCallback*

Description

Update the progress bar with input and prediction images

Usage

```
ShowCycleGANImgsCallback(imgA = FALSE, imgB = TRUE, show_img_interval = 10)
```

Arguments

imgA	img from A domain
imgB	img from B domain
show_img_interval	show image interval

Value

None

ShowGraphCallback *ShowGraphCallback*

Description

ShowGraphCallback

Usage

```
ShowGraphCallback(...)
```

Arguments

...	parameters to pass

Value

None

show_array *Show_array*

Description

Show an array on ‘ax‘.

Usage

```
show_array(
  array,
  ax = NULL,
  figsize = NULL,
  title = NULL,
  ctx = NULL,
  tx = NULL
)
```

Arguments

array	R array
ax	axis
figsize	figure size
title	title, text
ctx	ctx
tx	tx

Value

None

Examples

```
## Not run:

arr = as.array(1:10)
show_array(arr,title = 'My R array') %>% plot(dpi = 200)


## End(Not run)
```

---

show_batch *Show_batch*

Description

Show_batch

Usage

```
show_batch(
  dls,
  b = NULL,
  max_n = 9,
  ctxs = NULL,
  figsize = c(19.2, 10.8),
  show = TRUE,
  unique = FALSE,
  dpi = 90,
  ...
)
```

Arguments

dls	dataloader object
b	defaults to one_batch
max_n	maximum images
ctxs	ctxs parameter
figsize	figure size
show	show or not

unique	unique images
dpi	dots per inch
...	additional arguments to pass

Value

None

Examples

```
## Not run:

dls %>% show_batch()


## End(Not run)
```

---

show_image *show_image*

Description

Show a PIL or PyTorch image on ‘ax‘.

Usage

```
show_image(
  im,
  ax = NULL,
  figsize = NULL,
  title = NULL,
  ctx = NULL,
  cmap = NULL,
  norm = NULL,
  aspect = NULL,
  interpolation = NULL,
  alpha = NULL,
  vmin = NULL,
  vmax = NULL,
  origin = NULL,
  extent = NULL
)
```

Arguments

im	im
ax	axis
figsize	figure size
title	title
ctx	ctx
cmap	color maps
norm	normalization
aspect	aspect
interpolation	interpolation
alpha	alpha value
vmin	value min
vmax	value max
origin	origin
extent	extent

---

show_images *Show_images*

Description

Show all images ‘ims‘ as subplots with ‘rows‘ using ‘titles‘

Usage

```
show_images(
  ims,
  nrows = 1,
  ncols = NULL,
  titles = NULL,
  figsize = NULL,
  imsize = 3,
  add_vert = 0
)
```

Arguments

ims	images
nrows	number of rows
ncols	number of columns
titles	titles
figsize	figure size
imsize	image size
add_vert	add vertical

Value

None

`show_results` *Show_results*

Description

Show some predictions on ‘ds_idx‘-th dataset or ‘dl‘

Usage

```
show_results(
  object,
  ds_idx = 1,
  dl = NULL,
  max_n = 9,
  shuffle = TRUE,
  dpi = 90,
  ...
)
```

Arguments

`object`	model
`ds_idx`	ds by index
`dl`	DL application
`max_n`	maximum number of images
`shuffle`	shuffle or not
`dpi`	dots per inch
`...`	additional arguments

Value

None

`sigmoid` *Sigmoid*

Description

Same as ‘torch$sigmoid‘, plus clamping to ‘(eps,1-eps)

Usage

```
sigmoid(input, eps = 1e-07)
```

Arguments

`input`	inputs
`eps`	epsilon

Value

None

`SigmoidRange` *SigmoidRange*

Description

Sigmoid module with range ‘(low, high)‘

Usage

```
SigmoidRange(low, high)
```

Arguments

`low`	low value
`high`	high value

Value

None

sigmoid_ *Sigmoid_*

Description

Same as ‘torch$sigmoid_‘, plus clamping to ‘(eps,1-eps)

Usage

```
sigmoid_(input, eps = 1e-07)
```

Arguments

input	input
eps	eps

Value

None

sigmoid_range *Sigmoid_range*

Description

Sigmoid function with range ‘(low, high)‘

Usage

```
sigmoid_range(x, low, high)
```

Arguments

x	tensor
low	low value
high	high value

Value

None

SignalCutout *Signal Cutout*

Description

Randomly zeros some portion of the signal

Usage

```
SignalCutout(p = 0.5, max_cut_pct = 0.15)
```

Arguments

p	probability
max_cut_pct	max cut percentage

Value

None

SignalLoss *Signal Loss*

Description

Randomly loses some portion of the signal

Usage

```
SignalLoss(p = 0.5, max_loss_pct = 0.15)
```

Arguments

p	probability
max_loss_pct	max loss percentage

Value

None

SignalShifter *Signal Shifter*

Description

Randomly shifts the audio signal by ‘max_pct‘

Usage

```
SignalShifter(
  p = 0.5,
  max_pct = 0.2,
  max_time = NULL,
  direction = 0,
  roll = FALSE
)
```

Arguments

p	probability
max_pct	max percentage
max_time	maximum time
direction	direction
roll	roll or not

Details

direction must be -1(left) 0(bidirectional) or 1(right).

Value

None

SimpleCNN *SimpleCNN*

Description

Create a simple CNN with ‘filters‘.

Usage

```
SimpleCNN(filters, kernel_szs = NULL, strides = NULL, bn = TRUE)
```

Arguments

filters	filters number
kernel_szs	kernel size
strides	strides
bn	batch normalization

Value

None

SimpleSelfAttention *SimpleSelfAttention*

Description

Same as ‘nn$Module‘, but no need for subclasses to call ‘super()$__init__‘

Usage

```
SimpleSelfAttention(n_in, ks = 1, sym = FALSE)
```

Arguments

n_in	inputs
ks	kernel size
sym	sym

Value

None

---

sin.fastai.torch_core.TensorMask *Sin*

Description

Sin

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
sin(x)
```

Arguments

x	tensor

Value

tensor

`sinh.fastai.torch_core.TensorMask` *Sinh*

Description

Sinh

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
sinh(x)
```

Arguments

x	tensor

Value

tensor

`sin_` *Sin*

Description

Sin

Usage

```
## S3 method for class 'torch.Tensor'
sin(x)
```

Arguments

x	tensor

Value

tensor

skm_to_fastai *Skm to fastai*

Description

Convert ‘func‘ from sklearn$metrics to a fastai metric

Usage

```
skm_to_fastai(
  func,
  is_class = TRUE,
  thresh = NULL,
  axis = -1,
  activation = NULL,
  ...
)
```

Arguments

func	function
is_class	is classification or not
thresh	threshold point
axis	axis
activation	activation
...	additional arguments to pass

Value

None

slice *Slice*

Description

Slice

Usage

```
slice(...)
```

Arguments

...	additional arguments

Details

slice(start, stop[, step]) Create a slice object. This is used for extended slicing (e.g. a[0:10:2]).

Value

sliced object

`sort` *Sort*

Description

Sort

Usage

```
## S3 method for class 'torch.Tensor'
sort(x, decreasing = FALSE, ...)
```

Arguments

`x`	tensor
`decreasing`	the order
`...`	additional parameters to pass

`sort.fastai.torch_core.TensorMask` *Sort*

Description

Sort

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
sort(x, decreasing = FALSE, ...)
```

Arguments

`x`	tensor
`decreasing`	the order
`...`	additional parameters to pass

Value

tensor

SortedDL *SortedDL*

Description

A 'DataLoader' that goes throught the item in the order given by 'sort_func'

Usage

```
SortedDL(
  dataset,
  sort_func = NULL,
  res = NULL,
  bs = 64,
  shuffle = FALSE,
  num_workers = NULL,
  verbose = FALSE,
  do_setup = TRUE,
  pin_memory = FALSE,
  timeout = 0,
  batch_size = NULL,
  drop_last = FALSE,
  indexed = NULL,
  n = NULL,
  device = NULL
)
```

Arguments

dataset	dataset
sort_func	sort_func
res	res
bs	bs
shuffle	shuffle
num_workers	num_workers
verbose	verbose
do_setup	do_setup
pin_memory	pin_memory
timeout	timeout
batch_size	batch_size
drop_last	drop_last
indexed	indexed
n	n
device	device

Value

None

`SpacyTokenizer` *SpacyTokenizer*

Description

Spacy tokenizer for ‘lang‘

Usage

```
SpacyTokenizer(lang = "en", special_toks = NULL, buf_sz = 5000)
```

Arguments

`lang`	language
`special_toks`	special tokenizers
`buf_sz`	buffer size

Value

none

`SpearmanCorrCoef` *SpearmanCorrCoef*

Description

Spearman correlation coefficient for regression problem

Usage

```
SpearmanCorrCoef(
  dim_argmax = NULL,
  axis = 0,
  nan_policy = "propagate",
  activation = "no",
  thresh = NULL,
  to_np = FALSE,
  invert_arg = FALSE,
  flatten = TRUE
)
```

Arguments

dim_argmax	dim_argmax
axis	axis
nan_policy	nan_policy
activation	activation
thresh	thresh
to_np	to_np
invert_arg	invert_arg
flatten	flatten

Value

None

SpectrogramTransformer

Spectrogram Transformer

Description

Creates a factory for creating AudioToSpec

Usage

```
SpectrogramTransformer(mel = TRUE, to_db = TRUE)
```

Arguments

mel	mel-spectrogram or not
to_db	to decibels

Details

transforms with different parameters

Value

None

spec_add_spaces *Spec_add_spaces*

Description

Add spaces around / and #

Usage

```
spec_add_spaces(t)
```

Arguments

t	text

Value

string

sqrd *Sqrt*

Description

Sqrt

Usage

```
## S3 method for class 'torch.Tensor'
sqrt(x)
```

Arguments

x	tensor

Value

tensor

sqrt.fastai.torch_core.TensorMask *Sqrt*

Description

Sqrt

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
sqrt(x)
```

Arguments

x	tensor

Value

tensor

SqueezeNet *SqueezeNet*

Description

Base class for all neural network modules.

Usage

```
SqueezeNet(version = "1_0", num_classes = 1000)
```

Arguments

version	version of SqueezeNet
num_classes	the number of classes

Details

Your models should also subclass this class. Modules can also contain other Modules, allowing to nest them in a tree structure. You can assign the submodules as regular attributes:: import torch.nn as nn import torch.nn.functional as F class Model(nn.Module): def __init__(self): super(Model, self).__init__() self.conv1 = nn.Conv2d(1, 20, 5) self.conv2 = nn.Conv2d(20, 20, 5) def forward(self, x): x = F.relu(self.conv1(x)) return F.relu(self.conv2(x)) Submodules assigned in this way will be registered, and will have their parameters converted too when you call :meth:‘to‘, etc.

Value

model

`squeezenet1_0` *Squeezenet1_0*

Description

SqueezeNet model architecture from the '"SqueezeNet: AlexNet-level

Usage

```
squeezenet1_0(pretrained = FALSE, progress)
```

Arguments

`pretrained`	pretrained or not
`progress`	to see progress bar or not

Details

accuracy with 50x fewer parameters and <0.5MB model size" <https://arxiv.org/abs/1602.07360>'_ paper.

Value

model

`squeezenet1_1` *Squeezenet1_1*

Description

SqueezeNet 1.1 model from the 'official SqueezeNet repo

Usage

```
squeezenet1_1(pretrained = FALSE, progress)
```

Arguments

`pretrained`	pretrained or not
`progress`	to see progress bar or not

Details

<https://github.com/DeepScale/SqueezeNet/tree/master/SqueezeNet_v1.1>‘_. SqueezeNet 1.1 has 2.4x less computation and slightly fewer parameters than SqueezeNet 1.0, without sacrificing accuracy.

Value

model

---

stack_train_valid *Stack_train_valid*

Description

Stack df_train and df_valid, adds ‘valid_col‘=TRUE/FALSE for df_valid/df_train

Usage

```
stack_train_valid(df_train, df_valid)
```

Arguments

df_train	train data
df_valid	validation data

Value

data frame

---

step_stat *Step_stat*

Description

Register the number of steps done in ‘state‘ for ‘p‘

Usage

```
step_stat(p, step = 0, ...)
```

Arguments

p	p
step	step
...	additional args to pass

Value

None

---

sub *Sub*

Description

Sub

Usage

```
## S3 method for class 'torch.Tensor'
a - b
```

Arguments

a	tensor
b	tensor

Value

tensor

---

subplots *Subplots*

Description

Subplots

Usage

```
subplots(nrows = 2, ncols = 2, figsize = NULL, imsize = 4, add_vert = 0)
```

Arguments

nrows	number of rows
ncols	number of columns
figsize	figure size
imsize	image size
add_vert	add vertical

Value

plot object

sub_mask *Sub*

Description

Sub

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a - b
```

Arguments

a	tensor
b	tensor

Value

tensor

summary.fastai.learner.Learner *Summary*

Description

Summary

Usage

```
## S3 method for class 'fastai.learner.Learner'
summary(object, ...)
```

Arguments

object	model
...	additional arguments to pass

Value

None

Examples

```
## Not run:

summary(model)


## End(Not run)
```

`summary.fastai.tabular.learner.TabularLearner` *Summary*

Description

Print a summary of ‘m‘ using a output text width of ‘n‘ chars

Usage

```
## S3 method for class 'fastai.tabular.learner.TabularLearner'
summary(object, ...)
```

Arguments

`object`	model
`...`	additional parameters to pass

Value

None

`summary_plot` *Summary_plot*

Description

Displays the SHAP values (which can be interpreted for feature importance)

Usage

```
summary_plot(object, dpi = 200, ...)
```

Arguments

object	ShapInterpretation object
dpi	dots per inch
...	additional arguments

Value

None

---

swish *Swish*

Description

Swish

Usage

```
swish(x, inplace = FALSE)
```

Arguments

x	tensor
inplace	inplace or not

Value

None

Swish_ *Swish*

Description

Same as nn$Module, but no need for subclasses to call super()$__init__

Usage

```
Swish_(...)
```

Arguments

...	parameters to pass

Value

None

tabular *Tabular*

Description

Tabular

Usage

```
tabular()
```

Value

None

TabularDataTable *TabularDataTable*

Description

A 'Tabular' object with transforms

Usage

```
TabularDataTable(
  df,
  procs = NULL,
  cat_names = NULL,
  cont_names = NULL,
  y_names = NULL,
  y_block = NULL,
  splits = NULL,
  do_setup = TRUE,
  device = NULL,
  inplace = FALSE,
  reduce_memory = TRUE,
  ...
)
```

Arguments

df	A DataFrame of your data
procs	list of preprocess functions
cat_names	the names of the categorical variables
cont_names	the names of the continuous variables

y_names	the names of the dependent variables
y_block	the TransformBlock to use for the target
splits	How to split your data
do_setup	A parameter for if Tabular will run the data through the procs upon initialization
device	cuda or cpu
inplace	If True, Tabular will not keep a separate copy of your original DataFrame in memory
reduce_memory	fastai will attempt to reduce the overall memory usage
...	additional parameters to pass

Value

None

`TabularModel` *TabularModel*

Description

Basic model for tabular data.

Usage

```
TabularModel(
  emb_szs,
  n_cont,
  out_sz,
  layers,
  ps = NULL,
  embed_p = 0,
  y_range = NULL,
  use_bn = TRUE,
  bn_final = FALSE,
  bn_cont = TRUE,
  act_cls = nn$ReLU(inplace = TRUE)
)
```

Arguments

emb_szs	embedding size
n_cont	number of cont
out_sz	output size
layers	layers
ps	ps

embed_p	embed proportion
y_range	y range
use_bn	use batch normalization
bn_final	batch normalization final
bn_cont	batch normalization cont
act_cls	activation

Value

None

`TabularTS` *TabularTS*

Description

A 'DataFrame' wrapper that knows which cols are x/y, and returns rows in '__getitem__'

Usage

```
TabularTS(
  df,
  procs = NULL,
  x_names = NULL,
  y_names = NULL,
  block_y = NULL,
  splits = NULL,
  do_setup = TRUE,
  device = NULL,
  inplace = FALSE
)
```

Arguments

df	A DataFrame of your data
procs	list of preprocess functions
x_names	predictors names
y_names	the names of the dependent variables
block_y	the TransformBlock to use for the target
splits	How to split your data
do_setup	A parameter for if Tabular will run the data through the procs upon initialization
device	device name
inplace	If True, Tabular will not keep a separate copy of your original DataFrame in memory

Value

None

`TabularTSDataloader` *TabularTSDataloader*

Description

Transformed 'DataLoader'

Usage

```
TabularTSDataloader(
  dataset,
  bs = 16,
  shuffle = FALSE,
  after_batch = NULL,
  num_workers = 0,
  verbose = FALSE,
  do_setup = TRUE,
  pin_memory = FALSE,
  timeout = 0,
  batch_size = NULL,
  drop_last = FALSE,
  indexed = NULL,
  n = NULL,
  device = NULL
)
```

Arguments

`dataset`	data set
`bs`	batch size
`shuffle`	shuffle or not
`after_batch`	after batch
`num_workers`	the number of workers
`verbose`	verbose
`do_setup`	A parameter for if Tabular will run the data through the procs upon initialization
`pin_memory`	pin memory or not
`timeout`	timeout
`batch_size`	batch size
`drop_last`	drop last
`indexed`	indexed
`n`	n
`device`	device name

Value

None

`tabular_config` *Tabular_config*

Description

Convenience function to easily create a config for 'TabularModel'

Usage

```
tabular_config(
  ps = NULL,
  embed_p = 0,
  y_range = NULL,
  use_bn = TRUE,
  bn_final = FALSE,
  bn_cont = TRUE,
  act_cls = nn$ReLU(inplace = TRUE)
)
```

Arguments

`ps`	ps
`embed_p`	embed proportion
`y_range`	y_range
`use_bn`	use batch normalization
`bn_final`	batch normalization final
`bn_cont`	batch normalization
`act_cls`	activation

Value

None

`tabular_learner` *Tabular learner*

Description

Get a ‘Learner‘ using ‘dls‘, with ‘metrics‘, including a ‘TabularModel‘ created using the remaining params.

Usage

```
tabular_learner(
  dls,
  layers = NULL,
  emb_szs = NULL,
  config = NULL,
  n_out = NULL,
  y_range = NULL,
  loss_func = NULL,
  opt_func = Adam(),
  lr = 0.001,
  splitter = trainable_params(),
  cbs = NULL,
  metrics = NULL,
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE,
  moms = list(0.95, 0.85, 0.95)
)
```

Arguments

`dls`	It is a DataLoaders object.
`layers`	layers
`emb_szs`	emb_szs
`config`	config
`n_out`	n_out
`y_range`	y_range
`loss_func`	It can be any loss function you like.
`opt_func`	It will be used to create an optimizer when Learner.fit is called.
`lr`	It is learning rate.
`splitter`	It is a function that takes self.model and returns a list of parameter groups (or just one parameter group if there are no different parameter groups)

cbs	It is one or a list of Callbacks to pass to the Learner.
metrics	It is an optional list of metrics, that can be either functions or Metrics.
path	İt is used to save and/or load models.Often path will be inferred from dls, but you can override it or pass a Path object to model_dir. Make sure you can write in path/model_dir!
model_dir	İt is used to save and/or load models.Often path will be inferred from dls, but you can override it or pass a Path object to model_dir. Make sure you can write in path/model_dir!
wd	It is the default weight decay used when training the model.
wd_bn_bias	It controls if weight decay is applied to BatchNorm layers and bias.
train_bn	It controls if BatchNorm layers are trained even when they are supposed to be frozen according to the splitter.
moms	The default momentums used in Learner.fit_one_cycle.

Value

learner object

`tar_extract_at_filename` *Tar_extract_at_filename*

Description

Extract ‘fname‘ to ‘dest‘/‘fname.name‘ folder using ‘tarfile‘

Usage

```
tar_extract_at_filename(fname, dest)
```

Arguments

fname	folder name
dest	destination

Value

None

`tensor` *Tensor*

Description

Like ‘torch.as_tensor‘, but handle lists too, and can pass multiple vector elements directly.

Usage

```
tensor(...)
```

Arguments

`...`	image

Value

None

`TensorBBox` *TensorBBox*

Description

Basic type for a tensor of bounding boxes in an image

Usage

```
TensorBBox(x)
```

Arguments

`x`	tensor

Value

None

TensorBBox_create *TensorBBox_create*

Description

TensorBBox_create

Usage

```
TensorBBox_create(x, img_size = NULL)
```

Arguments

x	tensor
img_size	image size

Value

None

TensorImage *TensorImage*

Description

TensorImage

Usage

```
TensorImage(x)
```

Arguments

x	tensor

Value

None

TensorImageBW *TensorImageBW*

Description

TensorImageBW

Usage

```
TensorImageBW(x)
```

Arguments

x	tensor

Value

None

TensorMultiCategory *TensorMultiCategory*

Description

TensorMultiCategory

Usage

```
TensorMultiCategory(x)
```

Arguments

x	tensor

Value

None

TensorPoint *TensorPoint*

Description

Basic type for points in an image

Usage

```
TensorPoint(x)
```

Arguments

x	tensor

Value

None

TensorPoint_create *TensorPoint_create*

Description

Delegates (‘__call__‘,‘decode‘,‘setup‘) to (‘encodes‘,‘decodes‘,‘setups‘) if ‘split_idx‘ matches

Usage

```
TensorPoint_create(...)
```

Arguments

...	arguments to pass

Value

None

`TerminateOnNaNCallback` *TerminateOnNaNCallback*

Description

TerminateOnNaNCallback

Usage

```
TerminateOnNaNCallback(...)
```

Arguments

...	parameters to pass

Value

None

`test_loader` *Test_loader*

Description

Data loader. Combines a dataset and a sampler, and provides an iterable over

Usage

```
test_loader()
```

Details

the given dataset. The :class:‘~torch.utils.data.DataLoader‘ supports both map-style and iterable-style datasets with single- or multi-process loading, customizing loading order and optional automatic batching (collation) and memory pinning. See :py:mod:‘torch.utils.data‘ documentation page for more details.

Value

loader

text *Text module*

Description

Text module

Usage

```
text()
```

Value

None

TextBlock *TextBlock*

Description

A 'TransformBlock' for texts

Usage

```
TextBlock(
  tok_tfm,
  vocab = NULL,
  is_lm = FALSE,
  seq_len = 72,
  backwards = FALSE,
  min_freq = 3,
  max_vocab = 60000,
  special_toks = NULL,
  pad_tok = NULL
)
```

Arguments

tok_tfm	tok_tfm
vocab	vocab
is_lm	is_lm
seq_len	seq_len
backwards	backwards
min_freq	min_freq
max_vocab	max_vocab
special_toks	special_toks
pad_tok	pad_tok

Value

block object

`TextBlock_from_df` *TextBlock_from_df*

Description

Build a ‘TextBlock‘ from a dataframe using ‘text_cols‘

Usage

```
TextBlock_from_df(
  text_cols,
  vocab = NULL,
  is_lm = FALSE,
  seq_len = 72,
  backwards = FALSE,
  min_freq = 3,
  max_vocab = 60000,
  tok = NULL,
  rules = NULL,
  sep = " ",
  n_workers = 6,
  mark_fields = NULL,
  res_col_name = "text"
)
```

Arguments

`text_cols`	text columns
`vocab`	vocabulary
`is_lm`	is_lm
`seq_len`	sequence length
`backwards`	backwards
`min_freq`	minimum frequency
`max_vocab`	max vocabulary
`tok`	tokenizer
`rules`	rules
`sep`	separator
`n_workers`	number workers
`mark_fields`	mark_fields
`res_col_name`	result column name

Value

None

`TextBlock_from_folder` *TextBlock_from_folder*

Description

Build a ‘TextBlock‘ from a ‘path‘

Usage

```
TextBlock_from_folder(
  path,
  vocab = NULL,
  is_lm = FALSE,
  seq_len = 72,
  backwards = FALSE,
  min_freq = 3,
  max_vocab = 60000,
  tok = NULL,
  rules = NULL,
  extensions = NULL,
  folders = NULL,
  output_dir = NULL,
  skip_if_exists = TRUE,
  output_names = NULL,
  n_workers = 6,
  encoding = "utf8"
)
```

Arguments

`path`	path
`vocab`	vocabualry
`is_lm`	is_lm
`seq_len`	sequence length
`backwards`	backwards
`min_freq`	minimum frequency
`max_vocab`	max vocabulary
`tok`	tokenizer
`rules`	rules
`extensions`	extensions

folders	folders
output_dir	output_dir
skip_if_exists	skip_if_exists
output_names	output_names
n_workers	number of workers
encoding	encoding

Value

None

TextDataLoaders_from_csv

TextDataLoaders_from_csv

Description

Create from ‘csv‘ file in ‘path/csv_fname‘

Usage

```
TextDataLoaders_from_csv(
  path,
  csv_fname = "labels.csv",
  header = "infer",
  delimiter = NULL,
  valid_pct = 0.2,
  seed = NULL,
  text_col = 0,
  label_col = 1,
  label_delim = NULL,
  y_block = NULL,
  text_vocab = NULL,
  is_lm = FALSE,
  valid_col = NULL,
  tok_tfm = NULL,
  seq_len = 72,
  backwards = FALSE,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL
)
```

Arguments

path	path
csv_fname	csv file name
header	header
delimiter	delimiter
valid_pct	valid_ation percentage
seed	random seed
text_col	text column
label_col	label column
label_delim	label separator
y_block	y_block
text_vocab	text vocabulary
is_lm	is_lm
valid_col	valid column
tok_tfm	tok_tfm
seq_len	seq_len
backwards	backwards
bs	batch size
val_bs	validation batch size
shuffle_train	shuffle train data
device	device

Value

text loader

`TextDataLoaders_from_df`

TextDataLoaders_from_df

Description

Create from ‘df‘ in ‘path‘ with ‘valid_pct‘ ’

Usage

```
TextDataLoaders_from_df(
  df,
  path = ".",
  valid_pct = 0.2,
  seed = NULL,
  text_col = 0,
  label_col = 1,
  label_delim = NULL,
  y_block = NULL,
  text_vocab = NULL,
  is_lm = FALSE,
  valid_col = NULL,
  tok_tfm = NULL,
  seq_len = 72,
  backwards = FALSE,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL
)
```

Arguments

df	df
path	path
valid_pct	validation percentage
seed	seed
text_col	text_col
label_col	label_col
label_delim	label_delim
y_block	y_block
text_vocab	text_vocab
is_lm	is_lm
valid_col	valid_col
tok_tfm	tok_tfm
seq_len	seq_len
backwards	backwards
bs	batch size
val_bs	validation batch size, if not specified then val_bs is the same as bs.
shuffle_train	shuffle_train
device	device

Value

text loader

`TextDataLoaders_from_folder` *TextDataLoaders_from_folder*

Description

Create from imagenet style dataset in ‘path‘ with ‘train‘ and ‘valid‘ subfolders (or provide ‘valid_pct‘)

Usage

```
TextDataLoaders_from_folder(
  path,
  train = "train",
  valid = "valid",
  valid_pct = NULL,
  seed = NULL,
  vocab = NULL,
  text_vocab = NULL,
  is_lm = FALSE,
  tok_tfm = NULL,
  seq_len = 72,
  backwards = FALSE,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL
)
```

Arguments

`path`	path
`train`	train data
`valid`	validation data
`valid_pct`	validation percentage
`seed`	random seed
`vocab`	vocabulary
`text_vocab`	text_vocab
`is_lm`	is_lm
`tok_tfm`	tok_tfm
`seq_len`	seq_len

backwards	backwards
bs	batch size
val_bs	validation batch size
shuffle_train	shuffle train data
device	device

Value

text loader

`TextLearner` *TextLearner*

Description

Basic class for a 'Learner' in NLP.

Usage

```
TextLearner(
  dls,
  model,
  alpha = 2,
  beta = 1,
  moms = list(0.8, 0.7, 0.8),
  loss_func = NULL,
  opt_func = Adam,
  lr = 0.001,
  splitter = trainable_params(),
  cbs = NULL,
  metrics = NULL,
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE
)
```

Arguments

dls	dls
model	model
alpha	alpha
beta	beta
moms	moms

loss_func	loss_func
opt_func	opt_func
lr	lr
splitter	splitter
cbs	cbs
metrics	metrics
path	path
model_dir	model_dir
wd	wd
wd_bn_bias	wd_bn_bias
train_bn	train_bn

Value

None

`TextLearner_load_encoder` *Load_encoder*

Description

Load the encoder ‘file‘ from the model directory, optionally ensuring it’s on ‘device‘

Usage

```
TextLearner_load_encoder(file, device = NULL)
```

Arguments

file	file
device	device

Value

None

TextLearner_load_pretrained *Load_pretrained*

Description

Load a pretrained model and adapt it to the data vocabulary.

Usage

```
TextLearner_load_pretrained(wgts_fname, vocab_fname, model = NULL)
```

Arguments

wgts_fname	wgts_fname
vocab_fname	vocab_fname
model	model

Value

None

TextLearner_save_encoder *Save_encoder*

Description

Save the encoder to ‘file‘ in the model directory

Usage

```
TextLearner_save_encoder(file)
```

Arguments

file	file

Value

None

text_classifier_learner *Text_classifier_learner*

Description

Create a ‘Learner‘ with a text classifier from ‘dls‘ and ‘arch‘.

Usage

```
text_classifier_learner(
  dls,
  arch,
  seq_len = 72,
  config = NULL,
  backwards = FALSE,
  pretrained = TRUE,
  drop_mult = 0.5,
  n_out = NULL,
  lin_ftrs = NULL,
  ps = NULL,
  max_len = 1440,
  y_range = NULL,
  loss_func = NULL,
  opt_func = Adam,
  lr = 0.001,
  splitter = trainable_params,
  cbs = NULL,
  metrics = NULL,
  path = NULL,
  model_dir = "models",
  wd = NULL,
  wd_bn_bias = FALSE,
  train_bn = TRUE,
  moms = list(0.95, 0.85, 0.95)
)
```

Arguments

dls	dls
arch	arch
seq_len	seq_len
config	config
backwards	backwards
pretrained	pretrained

drop_mult	drop_mult
n_out	n_out
lin_ftrs	lin_ftrs
ps	ps
max_len	max_len
y_range	y_range
loss_func	loss_func
opt_func	opt_func
lr	lr
splitter	splitter
cbs	cbs
metrics	metrics
path	path
model_dir	model_dir
wd	wd
wd_bn_bias	wd_bn_bias
train_bn	train_bn
moms	moms

Value

None

TfmdDL *TfmdDL*

Description

Transformed ‘DataLoader‘

Usage

```
TfmdDL(
  dataset,
  bs = 64,
  shuffle = FALSE,
  num_workers = NULL,
  verbose = FALSE,
  do_setup = TRUE,
  pin_memory = FALSE,
  timeout = 0,
  batch_size = NULL,
```

```
  drop_last = FALSE,
  indexed = NULL,
  n = NULL,
  device = NULL,
  after_batch = NULL,
  ...
)
```

Arguments

dataset	dataset
bs	batch size
shuffle	shuffle
num_workers	number of workers
verbose	verbose
do_setup	do setup
pin_memory	pin memory
timeout	timeout
batch_size	batch size
drop_last	drop last
indexed	indexed
n	int, n
device	device
after_batch	after_batch
...	additional arguments to pass

Value

None

`TfmdLists` *TfmdLists*

Description

A ‘Pipeline‘ of ‘tfms‘ applied to a collection of ‘items‘

Usage

```
TfmdLists(...)
```

Arguments

...	parameters to pass

TfmResize *TfmResize*

Description

Temporary fix to allow image resizing transform

Usage

```
TfmResize(size, interp_mode = "bilinear")
```

Arguments

size	size
interp_mode	interpolation mode

Value

None

timm *Timm module*

Description

Timm module

Usage

```
timm()
```

Value

None

timm_learner *Timm_learner*

Description

Build a convnet style learner from ‘dls‘ and ‘arch‘ using the ‘timm‘ library

Usage

```
timm_learner(dls, arch, ...)
```

Arguments

dls	dataloader
arch	model architecture
...	additional arguments

Value

None

timm_list_models *Timm models*

Description

Timm models

Usage

```
timm_list_models(...)
```

Arguments

...	parameters to pass

Value

vector

tms *Timeseries module*

Description

Timeseries module

Usage

```
tms()
```

Value

None

tokenize1 *Tokenize1*

Description

Call ‘TokenizeWithRules‘ with a single text

Usage

```
tokenize1(text, tok, rules = NULL, post_rules = NULL)
```

Arguments

text	text
tok	tok
rules	rules
post_rules	post_rules

Value

None

Tokenizer *Tokenizer*

Description

Provides a consistent ‘Transform‘ interface to tokenizers operating on ‘DataFrame‘s and folders

Usage

```
Tokenizer(
  tok,
  rules = NULL,
  counter = NULL,
  lengths = NULL,
  mode = NULL,
  sep = " "
)
```

Arguments

tok	tokenizer
rules	rules
counter	counter
lengths	lengths
mode	mode
sep	separator

Value

None

Tokenizer_from_df *Tokenizer_from_df*

Description

Tokenizer_from_df

Usage

```
Tokenizer_from_df(
  text_cols,
  tok = NULL,
  rules = NULL,
  sep = " ",
  n_workers = 6,
  mark_fields = NULL,
  res_col_name = "text"
)
```

Arguments

text_cols	text columns
tok	tokenizer
rules	special rules
sep	separator
n_workers	number of workers
mark_fields	mark fields
res_col_name	output column name

Value

None

TokenizeWithRules *TokenizeWithRules*

Description

A wrapper around ‘tok‘ which applies ‘rules‘, then tokenizes, then applies ‘post_rules‘

Usage

```
TokenizeWithRules(tok, rules = NULL, post_rules = NULL)
```

Arguments

tok	tokenizer
rules	rules
post_rules	post_rules

Value

None

tokenize_csv *Tokenize_csv*

Description

Tokenize texts in the ‘text_cols‘ of the csv ‘fname‘ in parallel using ‘n_workers‘

Usage

```
tokenize_csv(
  fname,
  text_cols,
  outname = NULL,
  n_workers = 4,
  rules = NULL,
  mark_fields = NULL,
  tok = NULL,
  header = "infer",
  chunksize = 50000
)
```

Arguments

fname	file name
text_cols	text columns
outname	outname
n_workers	numeber of workers
rules	rules
mark_fields	mark fields
tok	tokenizer
header	header
chunksize	chunk size

Value

None

tokenize_df *Tokenize_df*

Description

Tokenize texts in ‘df[text_cols]‘ in parallel using ‘n_workers‘

Usage

```
tokenize_df(
  df,
  text_cols,
  n_workers = 6,
  rules = NULL,
  mark_fields = NULL,
  tok = NULL,
  res_col_name = "text"
)
```

Arguments

df	data frame
text_cols	text columns
n_workers	number of workers
rules	rules
mark_fields	mark_fields
tok	tokenizer
res_col_name	res_col_name

Value

None

tokenize_files *Tokenize_files*

Description

Tokenize text ‘files‘ in parallel using ‘n_workers‘

Usage

```
tokenize_files(
  files,
  path,
  output_dir,
  output_names = NULL,
  n_workers = 6,
  rules = NULL,
  tok = NULL,
  encoding = "utf8",
  skip_if_exists = FALSE
)
```

Arguments

files	files
path	path
output_dir	output_dir
output_names	output_names
n_workers	n_workers
rules	rules
tok	tokenizer
encoding	encoding
skip_if_exists	skip_if_exists

Value

None

`tokenize_folder` *Tokenize_folder*

Description

Tokenize text files in ‘path‘ in parallel using ‘n_workers‘

Usage

```
tokenize_folder(
  path,
  extensions = NULL,
  folders = NULL,
  output_dir = NULL,
  skip_if_exists = TRUE,
```

```
  output_names = NULL,
  n_workers = 6,
  rules = NULL,
  tok = NULL,
  encoding = "utf8"
)
```

Arguments

path	path
extensions	extensions
folders	folders
output_dir	output_dir
skip_if_exists	skip_if_exists
output_names	output_names
n_workers	number of workers
rules	rules
tok	tokenizer
encoding	encoding

Value

None

---

tokenize_texts *Tokenize_texts*

Description

Tokenize ‘texts‘ in parallel using ‘n_workers‘

Usage

```
tokenize_texts(texts, n_workers = 6, rules = NULL, tok = NULL)
```

Arguments

texts	texts
n_workers	n_workers
rules	rules
tok	tok

Value

None

top_k_accuracy *Top_k_accuracy*

Description

Computes the Top-k accuracy (‘targ‘ is in the top ‘k‘ predictions of ‘inp‘)

Usage

```
top_k_accuracy(inp, targ, k = 5, axis = -1)
```

Arguments

inp	predictions
targ	targets
k	k
axis	axis

Value

None

Examples

```
## Not run:

loaders = loaders()

data = Data_Loaders(loaders['train'], loaders['valid'])$cuda()

model = nn$Sequential() +
  nn$Flatten() +
  nn$Linear(28L * 28L, 10L)
metrics = list(accuracy,top_k_accuracy)
learn = Learner(data, model, loss_func = F$cross_entropy, opt_func = Adam,
                metrics = metrics)


## End(Not run)
```

torch *Builtins module*

Description

Builtins module

Usage

```
torch()
```

Value

None

ToTensor *ToTensor*

Description

Convert item to appropriate tensor class

Usage

```
ToTensor(enc = NULL, dec = NULL, split_idx = NULL, order = NULL)
```

Arguments

enc	encoder
dec	decoder
split_idx	int, split by index
order	order

Value

None

to_bytes_format *To_bytes_format*

Description

Convert to bytes, default to PNG format

Usage

```
to_bytes_format(img, format = "png")
```

Arguments

img	image
format	format

Value

None

to_image *To_image*

Description

Convert a tensor or array to a PIL int8 Image

Usage

```
to_image(x)
```

Arguments

x	tensor

Value

None

to_matrix *To matrix*

Description

To matrix

Usage

```
to_matrix(obj, matrix = TRUE)
```

Arguments

obj	learner/model
matrix	bool, to R matrix

to_thumb *To_thumb*

Description

Same as ‘thumbnail‘, but uses a copy

Usage

```
to_thumb(img, h, w = NULL)
```

Arguments

img	image
h	height
w	width

Value

None

`TrackerCallback` *TrackerCallback*

Description

A ‘Callback‘ that keeps track of the best value in ‘monitor‘.

Usage

```
TrackerCallback(monitor = "valid_loss", comp = NULL, min_delta = 0)
```

Arguments

`monitor`	monitor the loss
`comp`	comp
`min_delta`	minimum delta

Value

None

`trainable_params` *Trainable_params*

Description

Return all trainable parameters of ‘m‘

Usage

```
trainable_params(m)
```

Arguments

`m`	trainable parameters

Value

None

TrainEvalCallback *TrainEvalCallback*

Description

TrainEvalCallback

Usage

```
TrainEvalCallback(...)
```

Arguments

...	parameters to pass

Value

None

train_loader *Train_loader*

Description

Data loader. Combines a dataset and a sampler, and provides an iterable over

Usage

```
train_loader()
```

Details

the given dataset. The :class:‘~torch.utils.data.DataLoader‘ supports both map-style and iterable-style datasets with single- or multi-process loading, customizing loading order and optional automatic batching (collation) and memory pinning.

Value

loader

Transform *Transform*

Description

Delegates (‘__call__‘,‘decode‘,‘setup‘) to (‘encodes‘,‘decodes‘,‘setups‘) if ‘split_idx‘ matches

Usage

```
Transform(enc = NULL, dec = NULL, split_idx = NULL, order = NULL)
```

Arguments

enc	encoder
dec	decoder
split_idx	split by index
order	order

Value

None

TransformBlock *TransformBlock*

Description

A basic wrapper that links defaults transforms for the data block API

Usage

```
TransformBlock(
  type_tfms = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  dl_type = NULL,
  dls_kwargs = NULL
)
```

Arguments

type_tfms	transformation type
item_tfms	item transformation type
batch_tfms	one or several transforms applied to the batches once they are formed
dl_type	DL application
dls_kwargs	additional arguments

Value

block

`TransformersDropOutput` *TransformersDropOutput*

Description

TransformersDropOutput

Usage

```
TransformersDropOutput()
```

Value

None

`TransformersTokenizer` *TransformersTokenizer*

Description

TransformersTokenizer

Usage

```
TransformersTokenizer(tokenizer)
```

Arguments

`tokenizer`	tokenizer object

Value

None

trunc_normal_ *Trunc_normal_*

Description

Truncated normal initialization (approximation)

Usage

```
trunc_normal_(x, mean = 0, std = 1)
```

Arguments

x	tensor
mean	mean
std	standard deviation

Value

tensor

TSBlock *TSBlock*

Description

A TimeSeries Block to process one timeseries

Usage

```
TSBlock(...)
```

Arguments

...	parameters to pass

Value

None

TSDataLoaders_from_dfs

TSDataLoaders_from_dfs

Description

Create a DataLoader from a df_train and df_valid

Usage

```
TSDataLoaders_from_dfs(
  df_train,
  df_valid,
  path = ".",
  x_cols = NULL,
  label_col = NULL,
  y_block = NULL,
  item_tfms = NULL,
  batch_tfms = NULL,
  bs = 64,
  val_bs = NULL,
  shuffle_train = TRUE,
  device = NULL
)
```

Arguments

df_train	train data
df_valid	validation data
path	path (optional)
x_cols	predictors
label_col	label/output column
y_block	y_block
item_tfms	item transformations
batch_tfms	batch transformations
bs	batch size
val_bs	validation batch size
shuffle_train	shuffle train data
device	device name

Value

None

`TSDataTable` *TSDataTable*

Description

A 'DataFrame' wrapper that knows which cols are x/y, and returns rows in '__getitem__'

Usage

```
TSDataTable(
  df,
  procs = NULL,
  x_names = NULL,
  y_names = NULL,
  block_y = NULL,
  splits = NULL,
  do_setup = TRUE,
  device = NULL,
  inplace = FALSE
)
```

Arguments

`df`	A DataFrame of your data
`procs`	list of preprocess functions
`x_names`	predictors names
`y_names`	the names of the dependent variables
`block_y`	the TransformBlock to use for the target
`splits`	How to split your data
`do_setup`	A parameter for if Tabular will run the data through the procs upon initialization
`device`	device name
`inplace`	If True, Tabular will not keep a separate copy of your original DataFrame in memory

Value

None

`TSeries` *TSeries*

Description

Basic Time series wrapper

Usage

```
TSeries(...)
```

Arguments

`...`	parameters to pass

Value

None

`TSeries_create` *TSeries_create*

Description

TSeries_create

Usage

```
TSeries_create(x, ...)
```

Arguments

`x`	tensor
`...`	additional parameters

Value

tensor

Examples

```
## Not run:

res = TSeries_create(as.array(runif(100)))
res %>% show(title = 'R array') %>% plot(dpi = 200)


## End(Not run)
```

UnetBlock *UnetBlock*

Description

A quasi-UNet block, using 'PixelShuffle_ICNR upsampling'.

Usage

```
UnetBlock(
  up_in_c,
  x_in_c,
  hook,
  final_div = TRUE,
  blur = FALSE,
  act_cls = nn()$ReLU,
  self_attention = FALSE,
  init = nn()$init$kaiming_normal_,
  norm_type = NULL,
  ks = 3,
  stride = 1,
  padding = NULL,
  bias = NULL,
  ndim = 2,
  bn_1st = TRUE,
  transpose = FALSE,
  xtra = NULL,
  bias_std = 0.01,
  dilation = 1,
  groups = 1,
  padding_mode = "zeros"
)
```

Arguments

up_in_c	up_in_c parameter
x_in_c	x_in_c parameter
hook	The hook is set to this intermediate layer to store the output needed for this block.
final_div	final div
blur	blur is used to avoid checkerboard artifacts at each layer.
act_cls	activation
self_attention	self_attention determines if we use a self-attention layer
init	initializer
norm_type	normalization type
ks	kernel size
stride	stride
padding	padding mode
bias	bias
ndim	number of dimensions
bn_1st	batch normalization 1st
transpose	transpose
xtra	xtra
bias_std	bias standard deviation
dilation	dilation
groups	groups
padding_mode	The mode of padding

Value

None

---

unet_config *Unet_config*

Description

Convenience function to easily create a config for 'DynamicUnet'

Usage

```
unet_config(
  blur = FALSE,
  blur_final = TRUE,
  self_attention = FALSE,
  y_range = NULL,
  last_cross = TRUE,
  bottle = FALSE,
  act_cls = nn()$ReLU,
  init = nn()$init$kaiming_normal_,
  norm_type = NULL
)
```

Arguments

blur	blur is used to avoid checkerboard artifacts at each layer.
blur_final	blur final is specific to the last layer.
self_attention	self_attention determines if we use a self attention layer at the third block before the end.
y_range	If y_range is passed, the last activations go through a sigmoid rescaled to that range.
last_cross	last cros
bottle	bottle
act_cls	activation
init	initializer
norm_type	normalization type

Value

None

---

unet_learner *Unet_learner*

Description

Build a unet learner from ‘dls‘ and ‘arch‘

Usage

```
unet_learner(dls, arch, ...)
```

Arguments

dls	dataloader
arch	architecture
...	additional arguments

Value

None

---

uniform_blur2d *Uniform_blur2d*

Description

Uniformly apply blurring

Usage

```
uniform_blur2d(x, s)
```

Arguments

x	image
s	effect

Value

None

---

upit *Upit module*

Description

Upit module

Usage

```
upit()
```

Value

None

URLs_ADULT_SAMPLE *ADULT_SAMPLE dataset*

Description

download ADULT_SAMPLE dataset

Usage

```
URLs_ADULT_SAMPLE(filename = "ADULT_SAMPLE", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

Examples

```
## Not run:

URLs_ADULT_SAMPLE()


## End(Not run)
```

URLs_AG_NEWS *AG_NEWS dataset*

Description

download AG_NEWS dataset

Usage

```
URLs_AG_NEWS(filename = "AG_NEWS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

Examples

```
## Not run:

URLs_AG_NEWS()


## End(Not run)
```

`URLs_AMAZON_REVIEWSAMAZON_REVIEWS` *AMAZON_REVIEWSAMAZON_REVIEWS dataset*

Description

download AMAZON_REVIEWSAMAZON_REVIEWS dataset

Usage

```
URLs_AMAZON_REVIEWSAMAZON_REVIEWS(
  filename = "AMAZON_REVIEWSAMAZON_REVIEWS",
  untar = TRUE
)
```

Arguments

`filename`	the name of the file
`untar`	logical, whether to untar the '.tgz' file

Value

None

URLs_AMAZON_REVIEWS_POLARITY *AMAZON_REVIEWS_POLARITY dataset*

Description

download AMAZON_REVIEWS_POLARITY dataset

Usage

```
URLs_AMAZON_REVIEWS_POLARITY(
  filename = "AMAZON_REVIEWS_POLARITY",
  untar = TRUE
)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_BIWI_HEAD_POSE *BIWI_HEAD_POSE dataset*

Description

download BIWI_HEAD_POSE dataset

Usage

```
URLs_BIWI_HEAD_POSE(filename = "BIWI_HEAD_POSE", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_CALTECH_101 *CALTECH_101 dataset*

Description

download CALTECH_101 dataset

Usage

```
URLs_CALTECH_101(filename = "CALTECH_101", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_CAMVID *CAMVID dataset*

Description

download CAMVID dataset

Usage

```
URLs_CAMVID(filename = "CAMVID", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_CAMVID_TINY *CAMVID_TINY dataset*

Description

download CAMVID_TINY dataset

Usage

```
URLs_CAMVID_TINY(filename = "CAMVID_TINY", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_CARS *CARS dataset*

Description

download CARS dataset

Usage

```
URLs_CARS(filename = "CARS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_CIFAR *CIFAR dataset*

Description

download CIFAR dataset

Usage

```
URLs_CIFAR(filename = "CIFAR", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_CIFAR_100 *CIFAR_100 dataset*

Description

download CIFAR_100 dataset

Usage

```
URLs_CIFAR_100(filename = "CIFAR_100", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_COCO_TINY *COCO_TINY dataset*

Description

download COCO_TINY dataset

Usage

```
URLs_COCO_TINY(filename = "COCO_TINY", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_CUB_200_2011 *CUB_200_2011 dataset*

Description

download CUB_200_2011 dataset

Usage

```
URLs_CUB_200_2011(filename = "CUB_200_2011", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_DBPEDIA *DBPEDIA dataset*

Description

download DBPEDIA dataset

Usage

```
URLs_DBPEDIA(filename = "DBPEDIA", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_DOGS *DOGS dataset*

Description

download DOGS dataset

Usage

```
URLs_DOGS(filename = "DOGS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_FLOWERS *FLOWERS dataset*

Description

download FLOWERS dataset

Usage

```
URLs_FLOWERS(filename = "FLOWERS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_FOOD *FOOD dataset*

Description

download FOOD dataset

Usage

```
URLs_FOOD(filename = "FOOD", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_HORSE_2_ZEBRA *HORSE_2_ZEBRA dataset*

Description

download HORSE_2_ZEBRA dataset

Usage

```
URLs_HORSE_2_ZEBRA(filename = "horse2zebra", unzip = TRUE)
```

Arguments

filename	the name of the file
unzip	logical, whether to unzip the '.zip' file

Value

None

URLs_HUMAN_NUMBERS *HUMAN_NUMBERS dataset*

Description

download HUMAN_NUMBERS dataset

Usage

```
URLs_HUMAN_NUMBERS(filename = "HUMAN_NUMBERS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_IMAGENETTE *IMAGENETTE dataset*

Description

download IMAGENETTE dataset

Usage

```
URLs_IMAGENETTE(filename = "IMAGENETTE", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_IMAGENETTE_160 *IMAGENETTE_160 dataset*

Description

download IMAGENETTE_160 dataset

Usage

```
URLs_IMAGENETTE_160(filename = "IMAGENETTE_160", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_IMAGENETTE_320 *IMAGENETTE_320 dataset*

Description

download IMAGENETTE_320 dataset

Usage

```
URLs_IMAGENETTE_320(filename = "IMAGENETTE_320", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_IMAGEWOOF *IMAGEWOOF dataset*

Description

download IMAGEWOOF dataset

Usage

```
URLs_IMAGEWOOF(filename = "IMAGEWOOF", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_IMAGEWOOF_160 *IMAGEWOOF_160 dataset*

Description

download IMAGEWOOF_160 dataset

Usage

```
URLs_IMAGEWOOF_160(filename = "IMAGEWOOF_160", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_IMAGEWOOF_320 *IMAGEWOOF_320 dataset*

Description

download IMAGEWOOF_320 dataset

Usage

```
URLs_IMAGEWOOF_320(filename = "IMAGEWOOF_320", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_IMDB *IMDB dataset*

Description

download IMDB dataset

Usage

```
URLs_IMDB(filename = "IMDB", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_IMDB_SAMPLE *IMDB_SAMPLE dataset*

Description

download IMDB_SAMPLE dataset

Usage

```
URLs_IMDB_SAMPLE(filename = "IMDB_SAMPLE", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_LSUN_BEDROOMS *LSUN_BEDROOMS dataset*

Description

download LSUN_BEDROOMS dataset

Usage

```
URLs_LSUN_BEDROOMS(filename = "LSUN_BEDROOMS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_ML_SAMPLE *ML_SAMPLE dataset*

Description

download ML_SAMPLE dataset

Usage

```
URLs_ML_SAMPLE(filename = "ML_SAMPLE", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_MNIST *MNIST dataset*

Description

download MNIST dataset

Usage

```
URLs_MNIST(filename = "MNIST", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_MNIST_SAMPLE *MNIST_SAMPLE dataset*

Description

download MNIST_SAMPLE dataset

Usage

```
URLs_MNIST_SAMPLE(filename = "MNIST_SAMPLE", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_MNIST_TINY *MNIST_TINY dataset*

Description

download MNIST_TINY dataset

Usage

```
URLs_MNIST_TINY(filename = "MNIST_TINY", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_MNIST_VAR_SIZE_TINY *MNIST_VAR_SIZE_TINY dataset*

Description

download MNIST_VAR_SIZE_TINY dataset

Usage

```
URLs_MNIST_VAR_SIZE_TINY(filename = "MNIST_VAR_SIZE_TINY", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_MOVIE_LENS_ML_100k *MOVIE_LENS_ML_100k dataset*

Description

download MOVIE_LENS_ML_100k dataset

Usage

```
URLs_MOVIE_LENS_ML_100k(filename = "ml-100k", unzip = TRUE)
```

Arguments

filename	the name of the file
unzip	logical, whether to unzip the '.zip' file

Value

None

URLs_MT_ENG_FRA *MT_ENG_FRA dataset*

Description

download MT_ENG_FRA dataset

Usage

```
URLs_MT_ENG_FRA(filename = "MT_ENG_FRA", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_OPENAI_TRANSFORMER *OPENAI_TRANSFORMER dataset*

Description

download OPENAI_TRANSFORMER dataset

Usage

```
URLs_OPENAI_TRANSFORMER(filename = "OPENAI_TRANSFORMER", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_PASCAL_2007 *PASCAL_2007 dataset*

Description

download PASCAL_2007 dataset

Usage

```
URLs_PASCAL_2007(filename = "PASCAL_2007", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_PASCAL_2012 *PASCAL_2012 dataset*

Description

download PASCAL_2012 dataset

Usage

```
URLs_PASCAL_2012(filename = "PASCAL_2012", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_PETS *PETS dataset*

Description

download PETS dataset

Usage

```
URLs_PETS(filename = "PETS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_PLANET_SAMPLE *PLANET_SAMPLE dataset*

Description

download PLANET_SAMPLE dataset

Usage

```
URLs_PLANET_SAMPLE(filename = "PLANET_SAMPLE", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_PLANET_TINY *PLANET_TINY dataset*

Description

download PLANET_TINY dataset

Usage

```
URLs_PLANET_TINY(filename = "PLANET_TINY", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_S3_COCO *S3_COCO dataset*

Description

download S3_COCO dataset

Usage

```
URLs_S3_COCO(filename = "S3_COCO", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_S3_IMAGE *S3_IMAGE dataset*

Description

download S3_IMAGE dataset

Usage

```
URLs_S3_IMAGE(filename = "S3_IMAGE", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_S3_IMAGELOC *S3_IMAGELOC dataset*

Description

download S3_IMAGELOC dataset

Usage

```
URLs_S3_IMAGELOC(filename = "S3_IMAGELOC", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_S3_MODEL *S3_MODEL dataset*

Description

download S3_MODEL dataset

Usage

```
URLs_S3_MODEL(filename = "S3_MODEL", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_S3_NLP *S3_NLP dataset*

Description

download S3_NLP dataset

Usage

```
URLs_S3_NLP(filename = "S3_NLP", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_SIIM_SMALL *SIIM_SMALL*

Description

download YELP_REVIEWS_POLARITY dataset

Usage

```
URLs_SIIM_SMALL(filename = "SIIM_SMALL", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_SKIN_LESION *SKIN_LESION dataset*

Description

download SKIN_LESION dataset

Usage

```
URLs_SKIN_LESION(filename = "SKIN_LESION", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_SOGOU_NEWS *SOGOU_NEWS dataset*

Description

download SOGOU_NEWS dataset

Usage

```
URLs_SOGOU_NEWS(filename = "SOGOU_NEWS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_SPEAKERS10 *SPEAKERS10 dataset*

Description

download SPEAKERS10 dataset

Usage

```
URLs_SPEAKERS10(filename = "SPEAKERS10", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

Examples

```
## Not run:

URLs_SPEAKERS10()


## End(Not run)
```

URLs_SPEECHCOMMANDS *SPEECHCOMMANDS dataset*

Description

download SPEECHCOMMANDS dataset

Usage

```
URLs_SPEECHCOMMANDS(filename = "SPEECHCOMMANDS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

Examples

```
## Not run:

URLs_SPEECHCOMMANDS()


## End(Not run)
```

URLs_WIKITEXT *WIKITEXT dataset*

Description

download WIKITEXT dataset

Usage

```
URLs_WIKITEXT(filename = "WIKITEXT", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_WIKITEXT_TINY *WIKITEXT_TINY dataset*

Description

download WIKITEXT_TINY dataset

Usage

```
URLs_WIKITEXT_TINY(filename = "WIKITEXT_TINY", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_WT103_BWD *WT103_BWD dataset*

Description

download WT103_BWD dataset

Usage

```
URLs_WT103_BWD(filename = "WT103_BWD", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_WT103_FWD *WT103_FWD dataset*

Description

download WT103_FWD dataset

Usage

```
URLs_WT103_FWD(filename = "WT103_FWD", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_YAHOO_ANSWERS *YAHOO_ANSWERS dataset*

Description

download YAHOO_ANSWERS dataset

Usage

```
URLs_YAHOO_ANSWERS(filename = "YAHOO_ANSWERS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_YELP_REVIEWS *YELP_REVIEWS dataset*

Description

download YELP_REVIEWS dataset

Usage

```
URLs_YELP_REVIEWS(filename = "YELP_REVIEWS", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

URLs_YELP_REVIEWS_POLARITY *YELP_REVIEWS_POLARITY dataset*

Description

download YELP_REVIEWS_POLARITY dataset

Usage

```
URLs_YELP_REVIEWS_POLARITY(filename = "YELP_REVIEWS_POLARITY", untar = TRUE)
```

Arguments

filename	the name of the file
untar	logical, whether to untar the '.tgz' file

Value

None

vgg11_bn *Vgg11_bn*

Description

VGG 11-layer model (configuration "A") with batch normalization

Usage

```
vgg11_bn(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Very Deep Convolutional Networks For Large-Scale Image Recognition" <https://arxiv.org/pdf/1409.1556.pdf>

Value

model

vgg13_bn *Vgg13_bn*

Description

VGG 13-layer model (configuration "B") with batch normalization

Usage

```
vgg13_bn(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Very Deep Convolutional Networks For Large-Scale Image Recognition" <https://arxiv.org/pdf/1409.1556.pdf>

Value

model

vgg16_bn *Vgg16_bn*

Description

VGG 16-layer model (configuration "D") with batch normalization

Usage

```
vgg16_bn(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Very Deep Convolutional Networks For Large-Scale Image Recognition" <https://arxiv.org/pdf/1409.1556.pdf>

Value

model

vgg19_bn *Vgg19_bn*

Description

VGG 19-layer model (configuration 'E') with batch normalization

Usage

```
vgg19_bn(pretrained = FALSE, progress)
```

Arguments

pretrained	pretrained or not
progress	to see progress bar or not

Details

"Very Deep Convolutional Networks For Large-Scale Image Recognition" <https://arxiv.org/pdf/1409.1556.pdf>

Value

model

vision *Vision module*

Description

Vision module

Usage

```
vision()
```

Value

None

vleaky_relu *Vleaky_relu*

Description

'F$leaky_relu' with 0.3 slope

Usage

```
vleaky_relu(input, inplace = TRUE)
```

Arguments

input	inputs
inplace	inplace or not

Value

None

Voice *Voice*

Description

Voice

Usage

```
Voice(
  sample_rate = 16000,
  n_fft = 1024,
  win_length = NULL,
  hop_length = 128,
  f_min = 50,
  f_max = 8000,
  pad = 0,
  n_mels = 128,
  window_fn = torch()$hann_window,
  power = 2,
  normalized = FALSE,
  wkwargs = NULL,
  mel = TRUE,
  to_db = TRUE
)
```

Arguments

sample_rate	sample rate
n_fft	number of fast fourier transforms
win_length	windowing length
hop_length	hopping length
f_min	minimum frequency
f_max	maximum frequency
pad	padding mode
n_mels	number of mel-spectrograms
window_fn	window function
power	power
normalized	normalized or not
wkwargs	additional arguments
mel	mel-spectrogram or not
to_db	to decibels

Value

None

---

wandb *Wandb module*

Description

Wandb module

Usage

```
wandb()
```

Value

None

WandbCallback *WandbCallback*

Description

Saves model topology, losses & metrics

Usage

```
WandbCallback(
  log = "gradients",
  log_preds = TRUE,
  log_model = TRUE,
  log_dataset = FALSE,
  dataset_name = NULL,
  valid_dl = NULL,
  n_preds = 36,
  seed = 12345,
  reorder = TRUE
)
```

Arguments

log	"gradients" (default), "parameters", "all" or None. Losses & metrics are always logged.
log_preds	whether we want to log prediction samples (default to True).
log_model	whether we want to log our model (default to True). This also requires SaveModelCallback.
log_dataset	Options: - False (default) - True will log folder referenced by learn.dls.path. - a path can be defined explicitly to reference which folder to log. Note: subfolder "models" is always ignored.
dataset_name	name of logged dataset (default to folder name).
valid_dl	DataLoaders containing items used for prediction samples (default to random items from learn.dls.valid.
n_preds	number of logged predictions (default to 36).
seed	used for defining random samples.
reorder	reorder or not

Value

None

Warp *Warp*

Description

Apply perspective warping with ‘magnitude‘ and ‘p‘ on a batch of matrices

Usage

```
Warp(
  magnitude = 0.2,
  p = 0.5,
  draw_x = NULL,
  draw_y = NULL,
  size = NULL,
  mode = "bilinear",
  pad_mode = "reflection",
  batch = FALSE,
  align_corners = TRUE
)
```

Arguments

magnitude	magnitude
p	probability
draw_x	draw x
draw_y	draw y
size	size
mode	mode
pad_mode	padding mode
batch	batch
align_corners	align corners

Value

None

waterfall_plot *Waterfall_plot*

Description

Plots an explanation of a single prediction as a waterfall plot. Accepts a row_index and class_id.

Usage

```
waterfall_plot(object, row_idx = NULL, class_id = 0, dpi = 200, ...)
```

Arguments

object	ShapInterpretation object
row_idx	is the index of the row chosen in test_data to be analyzed, which defaults to zero.
class_id	Accepts a class_id which is used to indicate the class of interest for a classification model. It can either be an int or str representation for a class of choice.
dpi	dots per inch
...	additional arguments

Value

None

WeightDropout *WeightDropout*

Description

A module that wraps another layer in which some weights will be replaced by 0 during training.

Usage

```
WeightDropout(module, weight_p, layer_names = "weight_hh_l0")
```

Arguments

module	module
weight_p	weight_p
layer_names	layer_names

Value

None

weight_decay *Weight_decay*

Description

Weight decay as decaying ‘p‘ with ‘lr*wd‘

Usage

```
weight_decay(p, lr, wd, do_wd = TRUE, ...)
```

Arguments

p	p
lr	learning rate
wd	weight decay
do_wd	do_wd
...	additional args to pass

Value

None

Examples

```
## Not run:

tst_param = function(val, grad = NULL) {
  "Create a tensor with `val` and a gradient of `grad` for testing"
  res = tensor(val) %>% float()

  if(is.null(grad)) {
    grad = tensor(val / 10)
  } else {
    grad = tensor(grad)
  }

  res$grad = grad %>% float()
  res
}
p = tst_param(1., 0.1)
weight_decay(p, 1., 0.1)


## End(Not run)
```

win_abdoment_soft *Abdomen soft*

Description

Abdomen soft

Usage

```
win_abdoment_soft()
```

Value

list

win_brain *Brain*

Description

Brain

Usage

```
win_brain()
```

Value

list

win_brain_bone *Brain bone*

Description

Brain bone

Usage

```
win_brain_bone()
```

Value

list

win_brain_soft *Brain soft*

Description

Brain soft

Usage

```
win_brain_soft()
```

Value

list

win_liver *Liver*

Description

Liver

Usage

```
win_liver()
```

Value

list

win_lungs *Lungs*

Description

Lungs

Usage

```
win_lungs()
```

Value

list

win_mediastinum *Mediastinum*

Description

Mediastinum

Usage

```
win_mediastinum()
```

Value

list

win_spine_bone *Spine bone*

Description

Spine bone

Usage

```
win_spine_bone()
```

Value

list

win_spine_soft *Spine soft*

Description

Spine soft

Usage

```
win_spine_soft()
```

Value

list

win_stroke *Stroke*

Description

Stroke

Usage

```
win_stroke()
```

Value

list

win_subdural *Subdural*

Description

Subdural

Usage

```
win_subdural()
```

Value

list

XResNet *XResNet*

Description

A sequential container.

Usage

```
XResNet(block, expansion, layers, c_in = 3, c_out = 1000, ...)
```

Arguments

block	the blocks to pass to XResNet
expansion	argument for inputs and filters
layers	the layers to pass to XResNet
c_in	number of inputs
c_out	number of outputs
...	additional arguments

---

xresnet101 *Xresnet101*

Description

Load model architecture

Usage

```
xresnet101(...)
```

Arguments

...	parameters to pass

Value

model

---

xresnet152 *Xresnet152*

Description

Load model architecture

Usage

```
xresnet152(...)
```

Arguments

...	parameters to pass

Value

model

xresnet18 *Xresnet18*

Description

Load model architecture

Usage

```
xresnet18(...)
```

Arguments

...	parameters to pass

Value

model

xresnet18_deep *Xresnet18_deep*

Description

Load model architecture

Usage

```
xresnet18_deep(...)
```

Arguments

...	parameters to pass

Value

model

xresnet34 *Xresnet34*

Description

Load model architecture

Usage

```
xresnet34(...)
```

Arguments

...	parameters to pass

Value

model

xresnet34_deep *Xresnet34_deep*

Description

Load model architecture

Usage

```
xresnet34_deep(...)
```

Arguments

...	parameters to pass

Value

model

xresnet50 *Xresnet50*

Description

Load model architecture

Usage

```
xresnet50(...)
```

Arguments

...	parameters to pass

Value

model

xresnet50_deep *Xresnet50_deep*

Description

Load model architecture

Usage

```
xresnet50_deep(...)
```

Arguments

...	parameters to pass

Value

model

zoom *Zoom*

Description

Zoom

Usage

```
zoom(img, ratio)
```

Arguments

img	image files
ratio	ratio

Value

image

zoom_mat *Zoom_mat*

Description

Return a random zoom matrix with ‘max_zoom‘ and ‘p‘

Usage

```
zoom_mat(
  x,
  min_zoom = 1,
  max_zoom = 1.1,
  p = 0.5,
  draw = NULL,
  draw_x = NULL,
  draw_y = NULL,
  batch = FALSE
)
```

Arguments

x	tensor
min_zoom	minimum zoom
max_zoom	maximum zoom
p	probability
draw	draw
draw_x	draw x
draw_y	draw y
batch	batch

Value

None

&.fastai.torch_core.TensorMask
Logical_and

Description

Logical_and

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
x & y
```

Arguments

x	tensor
y	tensor

Value

tensor

%f% *Fastai assignment*

Description

The assignment has to be used for safe modification of the values inside tensors/layers

Usage

```
left %f% right
```

Arguments

left	left side object
right	right side object

Value

None

%%.fastai.torch_core.TensorMask *Floor mod*

Description

Floor mod

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
x %% y
```

Arguments

x	tensor
y	tensor

Value

tensor

`%/%.fastai.torch_core.TensorMask`
Floor divide

Description

Floor divide

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
x %/% y
```

Arguments

x	tensor
y	tensor

Value

tensor

`^.fastai.torch_core.TensorMask`
Pow

Description

Pow

Usage

```
## S3 method for class 'fastai.torch_core.TensorMask'
a ^ b
```

Arguments

a	tensor
b	tensor

Value

tensor

Index