Sat Nov 11 2017

utils.py

utils.py

cat()
copy_layer()
copy_layers()
copy_weights()
do_clip()
floor()
get_classes()
idxs_all()
limit_mem()
load_array()
plot()
plots()
save_array()
to_plot()
wrap_config()

Multerator

+ iters
+ multi
+ N

+ __init__()
+ reset()
+ __iter__()
+ next()

structured.py

structured.py

add_datepart(df, fldname, drop=True)
apply_cats(df, trn)
combine_date()
draw_tree(t, df, size=10, ratio=0.6, precision=0)
fix_missing(df, col, name, na_dict)
get_nn_mappers(df, cat_vars, contin_vars)
get_sample(df,n)
is_date(x)
mape_var_rmse(df, col, name, max_n_cat)
parallel_trees(m, fn, n_jobs=8)
proc_df()
reset_rf_samples()
rf_feat_importance(m, df)
scale_vars(df)
set_plot_sizes(sml, med, big)
set_rf_samples(n)
train_cats(df)

set_spawn.py

set_spawn.py

from multiprocessing import set_start_method
set_start_method('spawn')

plots.py

plots.py

plots()
plots_from_files()
plot_confusion_matrix()
plots_raw()
plot_train_val()
plot_val_with_title()
most_by_mask()
most_by_incorrect()

fastai

torch_imports.py

torch_imports.py

children(m)
save_model(m, p)
load_model(m, p)
load_pre(pre, f, fn)

inception_4(pre)
inceptionresnet_2(pre)

resnext50(pre)
resnext101(pre)
resnext101_64x4d(pre)
wrn(pre)

dn121(pre)
dn161(pre)
dn169(pre)
dn201(pre)

vgg16(pre)

imports.py

imports.py

in_notebook()
to(trn*args, **kwargs)
trange(*args, **kwargs)

cv2.setNumThreads(0)

core.py

core.py

apply_leaf(m, f)
chain_params(p)
noop(*args, **kwargs)
split_by_idx(idxs, a)
set_trainable(l, b)
set_trainable_attr(m,b)
SGD_Momentum(momentum)
split_by_idxs(seq, idxs)
sum_geom(a,r,n)
T(a)
to_gpu(x, *args, **kwargs)
to_np(v)
trainable_params_(m)
V_(x)
V(x)
VV_(x)
VV(x)

initializers.py

initializers.py

apply_init(m, init_fn)
cond_init(m, init_fn)

io

io.py

get_data(url, filename)

tqdm

tqdm

TqdmUpTo

+ total
+ update_to()

torch

torch.nn

torch.autograd

torch.utils

itertools

ipywidgets

torchvision

torchvision.models

torchtext

functools

math

IPython.lib.deepreload

PIL

numpy

cv2

collections

threading

scipy

pandas

seaborn

matplotlib

isoweek

json

pandas_summary

bcolz

IPython.lib.display

random

pickle

layer_optimizer.py

layer_optimizer.py

opt_params()
set_lrs()

LayerOptimizer

+ opt
+ lrs

+ __init__()
+ opt_params()
+ lr()
+ set_lrs()

layers.png

layers

AdaptiveConcatPool2d

+ sp
+ mp

+ __init__()
+ forward()

Flatten

+ __init__()
+ forward()

Lambda

+ f

+ __init__()
+ forward()

nn.Module

concurrent.futures

enum

sys

abc

glob

string

time

datetime

re

enum.IntEnum

transforms

transforms.py

transforms.py

imagenet_mean
transforms_basic
transforms_side_on
transforms_top_down

preprocess_imagenet(x)
preprocess_scale(x)
scale_min(im, targ)
zoom_cv(x,z)
stretch_cv(x,ar,ac)
dihedral(x, dih)
rotate_cv(x, deg, mode=cv2.BORDER_REFLECT, flags=cv2.INTER_LINEAR)
no_crop(im, min_sz=None)
center_crop(im, min_sz=None)
scale_to(x, ratio, targ)
crop(im, r, c, sz)
det_dihedral(dih)
det_stretch(sz, sc)
det_zoom(zoom)
det_rotate(deg)
det_zoom(zoom)
rand0(s)
CenterCrop(im, sz=None)
NoCrop(im, sz=None)
Scale(sz)
RandomFlip()
GetDet(det, y, i)
to_bb(YY, y)
coords2px(y, x)
random_px_rect(y, x)
compose(im, y, fns)
image_gen(normalizer, denorm, sz, tfms=None, max_zoom=None, pad=0, crop_type=None, tfm_y=None)
noop(x)

Transform

+ tfm_y

+ __init__()
+ __call__()
+ transform_coord()
+ transform()
+ do_transform()

Transforms

+ tfms

+ __init__()
+ __call__()

CropType

+ RANDOM
+ CENTER
+ NO

TfmType

+ NO
+ PIXEL
+ COORD

Normalize

+ m
+ s

+ __init__()
+ __call__()

Denormalize

+ m
+ s

+ __init__()
+ __call__()

RandomLighting

+ __init__()
+ __call__()

RandomLightingXY

+ tfm_y
+ b_rand
+ c_rand

+ __init__()
+ set_state()
+ do_transform()

RandomRotate

+ __init__()
+ __call__()

CoordTransform

+ transform_coord()

RandomRotateXY

+ tfm_y
+ rdeg
+ rp

+ __init__()
+ set_state()
+ transform_coord()
+ do_transform()
+ do_transform_y()

RandomRotateZoom

+ __init__()
+ __call__()

ReflectionPad

+ __init__()
+ add_pad()
+ __call__()

CenterCropXY

+ tfm_y
+ min_sz

+ __init__()
+ set_state()
+ do_transform()

NoCropXY

+ tfm_y
+ sz

+ __init__()
+ set_state()
+ do_transform()

RandomCropXY

+ tfm_y
+ targ
+ rand_r
+ rand_c

+ __init__()
+ set_state()
+ do_transform()

RandomDihedralXY

+ m
+ s

+ set_state()
+ do_transform()

RandomFlipXY

+ do_flip

+ set_state()
+ do_transform()

RandomScaleXY

+ new_sz

+ __init__()
+ set_state()
+ do_transform()

ScaleXY

+ tfm_y
+ szy

+ __init__()
+ set_state()
+ do_transform()

RandomCrop

+ targ

+ __init__()
+ __call__()

RandomDihedral

+ __call__()

RandomScale

+ __init__()
+ __call__()

model

model.py

model.py

cut_model()
predict_to_bcolz()
num_features()
set_train_mode()
fit()
validate()
get_prediction()
predict()
predict_with_targs()
model_summary()

Stepper

+ __init__()
+ reset()
+ step()
+ evaluate()

column_data

column_data.py

ModelData

...

Dataset

...

CollabFilterDataset

...

CollabFilterLearner

...

CollabFilterModel

...

BasicModel

...

ColumnarDataset

...

ColumnarModelData

...

EmbeddingDotBias

...

Learner

...

MixedInputModel

...

StructuredLearner

...

StructuredModel

...

nn.Module

...

create list of network architectures

Dataset

...

Learner

...

BasicModel

...

NLP

nlp.py

BOW_Dataset

...

BOW_Learner

...

TextModel

...

nn.Module

...

DotProdNB

...

LanguageModelData

...

LanguageModelLoader

...

SimpleNB

...

RNN_Learner

...

TextDataLoader

...

TextClassifierData

...

TextData

...

ConcatTextDataset

...

ModelData

...

torchtext.data.Dataset

models

wrn 50 2f

wrn_50_2f.py

wrn_50_2f.py

wrn_50_2f()

LambdaBase

+ lambda_func

+ __init__()
+ forward_prepare()

LambdaReduce

+ forward()

LambdaMap

+ forward()

Lambda

+ forward()

nn.Sequential

convert torch

convert_torch.py

convert_torch.py

add_submods(seq, *args)
copy_params(m,n)
lua_recursive_model(module,seq)
lua_recursive_source(module)
simplify_source(s)
torch_to_pytorch(7_filename,outputname=None)

LambdaBase

+ lambda_func

+ __init__()
+ forward_prepare()

LambdaReduce

+ forward()

LambdaMap

+ forward()

Lambda

+ forward()

nn.Module

ResNext 101 64x4d

resnext_101_64x4d.py

resnext_101_64x4d.py

resnext_101_64x4d()

LambdaBase

+ lambda_func

+ __init__()
+ forward_prepare()

LambdaReduce

+ forward()

LambdaMap

+ forward()

Lambda

+ forward()

ResNet 50 32x4d

resnext_50_32x4d.py

resnext_50_32x4d.py

resnext_50_32x4d()

LambdaBase

+ lambda_func

+ __init__()
+ forward_prepare()

LambdaReduce

+ forward()

LambdaMap

+ forward()

Lambda

+ forward()

ResNext 101 32x4d

resnext_101_32x4d.py

resnext_101_32x4d.py

resnext_101_32x4d()

LambdaBase

+ lambda_func

+ __init__()
+ forward_prepare()

LambdaReduce

+ forward()

LambdaMap

+ forward()

Lambda

+ forward()

torchvision.models

resnet.py

resnet18(pretrained=False, **kwargs)
resnet34(pretrained=False, **kwargs)
resnet50(pretrained=False, **kwargs)
resnet101(pretrained=False, **kwargs)

densenet.py

densenet121(pretrained=False, **kwargs)
densenet161(pretrained=False, **kwargs)
densenet169(pretrained=False, **kwargs)
densenet201(pretrained=False, **kwargs)

vgg.py

vgg16_bn(pretrained=False, **kwargs)

Inception ResNet v2

inceptionresnetv2.py

inceptionresnetv2.py

inceptionresnetv2(pretrained=True)
load()
test(model)

InceptionResnetV2

+ conv2d_1a
+ conv2d_2a
+ conv2d_2b
+ maxpool_3a
+ conv2d_3b
+ conv2d_4a
+ maxpool_5a
+ mixed_5b
+ repeat
+ mixed_6a
+ repeat_1
+ mixed_7a
+ repeat_2
+ block8
+ conv2d_7b
+ avgpool_1a
+ classif

+ __init__()
+ forward()

BasicConv2d

+ conv
+ bn
+ relu

+ __init__()
+ forward()

Mixed_6a

+ branch0
+ branch1
+ branch2

+ __init__()
+ forward()

Mixed_5b

+ branch0
+ branch1
+ branch2

+ __init__()
+ forward()

Mixed_7a

+ branch0
+ branch1
+ branch2
+ branch3

+ __init__()
+ forward()

Block8

+ scale
+ noReLU
+ branch0
+ branch1
+ conv2d
+ relu

+ __init__()
+ forward()

Block17

+ scale
+ branch0
+ branch1
+ conv2d
+ relu

+ __init__()
+ forward()

Block35

+ scale
+ branch0
+ branch1
+ branch2
+ conv2d
+ relu

+ __init__()
+ forward()

nn.Module

Inception v4

inceptionv4.py

inceptionv4.py

inceptionv4(pretrained=True)
load()
test(model)

InceptionV4

+ features
+ classif

+ __init__()
+ forward()

BasicConv2d

+ conv
+ bn
+ relu

+ __init__()
+ forward()

Inception_A

+ branch0
+ branch1
+ branch2
+ branch3

+ __init__()
+ forward()

Inception_B

+ branch0
+ branch1
+ branch2

+ __init__()
+ forward()

Inception_C

+ branch0
+ branch1_0
+ branch1_1a
+ branch1_1b
+ branch2_0
+ branch2_1
+ branch2_2
+ branch2_3a
+ branch2_3b
+ branch3

+ __init__()
+ forward()

Mixed_3a

+ maxpool
+ conv

+ __init__()
+ forward()

Mixed_4a

+ branch0
+ branch1

+ __init__()
+ forward()

Mixed_5a

+ maxpool
+ conv

+ __init__()
+ forward()

Reduction_A

+ branch0
+ branch1
+ branch2

+ __init__()
+ forward()

Reduction_B

+ branch0
+ branch1
+ branch2

+ __init__()
+ forward()

nn.Module

RNN

LM RNN

lm_rnn.py

nn.Module

...

nn.Sequential

...

RNN_Encoder

+ initrange
+ encoder
+ rnns
+ dropout
+ dropouth
+ hidden
+ weights

+ __init__()
+ forward()
+ one_hot(bsz)
+ reset()

LinearRNNOutput

+ initrange
+ decoder
+ dropout

+ __init__()
+ forward()

SequentialRNN

+ reset()

MultiBatchRNN

+ bs

+ __init__()
+ concat()
+ forward()

LinearDecoder

+ __init__()
+ forward()

PoolingLinearClassifier

+ pool()
+ forward()

rnn_train.py

imports only

RNN reg

rnn_reg.py

nn.Module

...

WeightDrop

+ __init__()
+ _setup()
+ _setweights()
+ forward()

LockedDropout

+ p

+ __init__()
+ forward()

rnn_reg.py

dropout_mask(x, sz, dropout)
embedded_dropout(embed, words, dropout=0.1, scale=None)

learner

learner.py

learner.py

Learner

clip
crit
data
metrics
models
models_path
opt_fn
reg_fn
sched
tmp_path

# properties
children()
model()
data()

__init__()
__getitem__()
summary()
freeze_to(n)
unfreeze()
get_model_path(name)
save(name)
load(name)
set_data(data)
get_cycle_end(name)
save_cycle(name, cycle)
load_cycle(name, cycle)

fit_gen(model, data, layer_opt, n_cycle, cycle_len=None, cycle_mult=1, cycle_save_name=None, metrics=None, callbacks=None, **kwargs)

get_layer_groups()
get_layer_opt(lrs, wds)
fit(lrs, n_cycle, wds=None, **kwargs)
lr_find(start_lr=1e-5, end_lr=10, wds=None)
predict(is_test=False)
predict_with_targs(is_test=False)
TTA(n_aug=4, is_test=False)

BasicModel

+ model

+ __init__()
+ get_layer_groups()

SingleModel

+ get_layer_groups()

conv learner

conv_learner.py

conv_learner.py

model_meta = {}
model_features = {}

ConvLearner

activations
crit
fc_data
metrics
precompute

# properties
model()
data()

__init__()
pretrained(f, data, pss=None, xtra_fc=None, xtra_cut=0, **kwargs)
create_empty_bcolz(n, name)
set_data(data)
get_layer_groups()
get_activations(force=False)
save_fc1()
freeze()

ConvnetBuilder

c
f
fc_model
ss_multi
is_reg
ps
top_model
xtra_cut
xtra_fc

name()

__init__()
create_fc_layer(ni, nf, p, actn=None)
get_fc_layers()
get_layer_groups(do_fc=False)

losses.py

losses.py

fbeta_torch(y_true, y_pred, beta, threshold, eps=1e-9)

metrics.py

metrics.py

accuracy(preds, targs)
accuracy_multi(preds, targs, thresh)
accuracy_thresh(thresh)

dataset

dataloader.py

dataloader.py

numpy_type_map

np_collate(batch)
get_tensor(batch, pin)

DataLoader

batch_sampler
batch_size
collate_fn
dataset
drop_last
num_workers
pin_memory
sampler

__init__(dataset, batch_size=1, shuffle=False, sampler=None, batch_sampler=None, num_workers=0, collate_fn=np_collate, pin_memory=False, drop_last=False)
__len__()
get_batch(indices)
__iter__()

dataset.py

dataset.py

imagenet_stats
inception_stats
inception_models

get_cv_idxs(n, cv_idx=4, val_pct=0.2, seed=42)
resize_img(fname, targ, path, new_path)
resize_imgs(fnames, targ, path, new_path)
read_dirs(path, folder)
read_dirs(path, folder)
n_hot(ids, c)
folder_source(path, folder)
parse_csv_labels(fn, skip_header=True)
nhot_labels(label2idx, csv_labels, fnames, c)
csv_source(folder, csv_file, skip_header=True, suffix='', continuous=False)
split_by_idx(idxs, *a)
tfms_from_model(f_model, sz, aug_tfms=[], max_zoom=None, pad=0, crop_type=None, tfm_y=None)

ModelDataLoader

dl
i
n

dataset()

__init__(dl)
__getattr__(k)(*args, **kwargs)
__iter__()
__len__()
__next__()

FilesDataset

fnames
path

__init__(fnames, transform, path)
get_n()
get_sz()
get_x()
resize_imgs(targ, new_path)
denorm(arr)

BaseDataset

c
n
sz
transform

is_multi()
is_reg()

__init__(transform=None)
__getitem__(idx)
__len__()
get(self, tfm, x, y)
get_n()
get_c()
get_sz()
get_x()
get_y()

torch.utils.data.dataset.Dataset

__getitem__(index)
__len__()

FilesArrayDataset

y

__init__(fnames, y, transform, path)
get_y(i)
get_c()

FilesIndexArrayDataset

get_c()

FilesIndexArrayRegressionDataset

is_reg()

FilesNhotArrayDataset

is_multi()

ModelData

path
test_dl
trn_dl
val_dl

is_reg()
trn_ds()
val_ds()
test_ds()
trn_y()
val_y()

__init__(path, trn_dl, val_dl, test_dl=None)
from_dls(cls, path,trn_dl,val_dl,test_dl=None)

ImageData

aug_dl
bs
classes
fix_dl
num_workers
path
test_aug_dl
test_dl
trn_dl
val_dl

sz()
c()

__init__(path, datasets, bs, num_workers, classes)
get_dls(cls, shuffle)
resize(dl, targ, new_path)
resize(targ, new_path)

ArraysDataset

x
y

__init__(x, y, transform)
get_x(i)
get_y(i)
get_n()
get_sz()

ArraysIndexDataset

get_c()

ArraysNhotDataset

is_multi()

get_c()

ImageClassifierData

is_multi()

get_dls(trn, val, tfms, test=None, **kwargs)
from_arrays(path, trn, val, bs=64, tfms=(None,None), classes=None, num_workers=4, test=None)
from_paths(path, bs=64, tfms=(None,None), trn_name='train', val_name='valid', test_name=None, num_workers=8)
from_csv(self, path, folder, csv_fname, bs=64, tfms=(None,None), val_idxs=None, suffix='', test_name=None, continuous=False, skip_header=True, num_workers=8)

SGDR

sgdr.py

sgdr.py

Callback

+ on_train_begin()
+ on_epoch_end()
+ on_batch_end()

LossRecorder

+ layer_opt
+ init_lrs
+ iterations
+ epoch

+ __init__()
+ on_train_begin()
+ on_epoch_end()
+ on_batch_end()
+ plot_loss()
+ plot_lr()

LR_Updater

+ on_train_begin()
+ on_batch_end()
+ update_lr()
+ calc_lr()

LR_Finder

+ lr_mult
+ best

+ __init__()
+ on_train_begin()
+ calc_lr()
+ on_batch_end()
+ plot()

CosAnneal

+ cycle_iter

+ __init__()
+ on_train_begin()
+ calc_lr()

```
# This file contains all the main external libs we'll use
from fastai.imports import *

from fastai.transforms import *
from fastai.conv_learner import *
from fastai.model import *
from fastai.dataset import *
from fastai.sgdr import *
from fastai.plots import *
```