NeuralGym Documentation

Release 0.0.1

Jiahui Yu

Dec 05, 2017

Tutorials

CHAPTER 1

Generative Adversarial Networks

CHAPTER 2

Image Classification

CHAPTER 3

Reinforcement Learning

CHAPTER 4

neuralgym

class neuralgym.**Config**(*filename=None*)

Bases: dict

Config with yaml file.

This class is used to config model hyper-parameters, global constants, and other settings with yaml file. All settings in yaml file will be automatically logged into file.

Parameters **filename** (*str*) – File name.

Examples

yaml file `model.yml`:

```
NAME: 'neuralgym'
ALPHA: 1.0
DATASET: '/mnt/data/imagenet'
```

Usage in .py:

```
>>> from neuralgym import Config
>>> config = Config('model.yml')
>>> print(config.NAME)
    neuralgym
>>> print(config.ALPHA)
    1.0
>>> print(config.DATASET)
    /mnt/data/imagenet
```

neuralgym.**get_gpus**(*num_gpus=1*, *dedicated=True*, *verbose=True*)

Auto-select gpus for running by setting CUDA_VISIBLE_DEVICES.

Parameters

- **num_gpus** (*int*) – Number of GPU(s) to get.

- **dedicated** (*bool*) – Dedicated GPU or not, i.e. one process for one GPU.
- **verbose** (*bool*) – Display nvidia-smi info if verbose is true.

Returns A list of selected GPU(s).

Return type list

neuralgym.**set_gpus**(*gpus*)
Set environment variable CUDA_VISIBLE_DEVICES to a list of gpus.

Parameters **gpus** (*int or list*) – GPU id or a list of GPU ids.

neuralgym.**date_uid**()
Generate a unique id based on date.

Returns Return uid string, e.g. '20171122171307111552'.

Return type str

neuralgym.**unset_logger**()
Unset logger of neuralgym.

CHAPTER 5

neuralgym.callbacks

5.1 Callback Class

class neuralgym.callbacks.**Callback**(*cb_loc*)

Callback class.

Callbacks are functions that execute automatically during training/evaluation process (primary trainer). For examples, saving/loading models, scheduling learning rate, updating network parameters (assigning target network in reinforcement learning), summary learning processes, saving images, secondary trainer (generative adversarial network), etc.

Currently there are three types of callbacks:

- *OnceCallback*
- *PeriodicCallback*
- *ScheduledCallback*

and five types of locations to call (supported in primary trainer):

- train_start
- train_end
- step_start
- step_end
- exception

run()

Abstract method for executing the callback.

Note: ops should be defined in __init__, otherwise when callbacks are called multiple times, the ops in graph will continue increasing.

class neuralgym.callbacks.**PeriodicCallback**(*cb_loc*, *pstep*, *func=None*, ***kwargs*)

PeriodicCallback executes periodically.

PeriodicalCallback is executed at:

1. at the step start, and
2. at the step end

of training every p steps periodically.

Parameters

- **cb_loc** – callback location
- **pstep** (*int*) – run function every pstep
- **func** (*function*) – function to call
- ****kwargs** – kwargs for function

class neuralgym.callbacks.**OnceCallback**(*cb_loc*, *func=None*, ***kwargs*)

OnceCallback only executes once.

OnceCallback is executed:

1. at the train start,
2. at the train end, and
3. when exception occurs

during training process.

class neuralgym.callbacks.**ScheduledCallback**(*cb_loc*, *schedule*)

ScheduledCallback executes according to its schedule.

ScheduledCallback is executed:

1. at the step start, and
2. at the step end

according to recorded step in schedule.

Parameters

- **cb_loc** – callback location
- **schedule** (*dict*) – a dict, with step as its key, funcs as its value: e.g. {1: func1, 80: func2}

5.2 Callbacks

5.2.1 HyperParamScheduler

class neuralgym.callbacks.**HyperParamScheduler**(*param_name*, *schedule*, *scope=None*, *cb_loc=<CallbackLoc.step_end: 3>*)

Set hyper parameters according to schedule.

This callback sets hyper parameters with numpy using tf.assign according to schedule.

Examples:

```
HyperParamScheduler(
    'lr',
    {
        1: 1e-2,
        150: 1e-3,
        225: 4e-4,
        300: 1e-4,
    },
    scope=None,
)
```

5.2.2 WeightsViewer

class neuralgym.callbacks.**WeightsViewer**(*counts=True*, *size=True*, *verbose=True*, *hist_summary=True*)

WeightsViewer logs names and size of all weights.

Parameters

- **counts** (*bool*) – Counting trainalbe weights or not.
- **size** (*bool*) – Size of trainable weights or not.
- **verbose** (*bool*) – Display each trainable variable or not.
- **hist_summary** (*bool*) – Histogram summary of trainable weights or not.

5.2.3 ModelSaver

class neuralgym.callbacks.**ModelSaver**(*pstep*, *saver*, *dump_prefix*)

Save model to file at every pstep step_start.

Parameters

- **pstep** (*int*) – Save to model every pstep.
- **saver** – Tensorflow saver.
- **dump_prefix** (*str*) – Prefix for saving model files.

5.2.4 ModelRestorer

class neuralgym.callbacks.**ModelRestorer**(*saver*, *dump_prefix=None*, *ckpt_file=None*, *optimistic=False*)

Restore model from file either with dump_prefix or ckpt_file.

Parameters

- **saver** – Tensorflow saver.
- **dump_prefix** (*str*) – Prefix of model files.
- **ckpt_file** (*str*) – Exact name of model file.
- **optimistic** (*bool*) – Only restore weights of same names with model.

5.2.5 NPZModelLoader

class neuralgym.callbacks.**NPZModelLoader**(*npz_file*, *weights=None*, *variable_scope=<tensorflow.python.ops.variable_scope.VariableScope object>*)

NPZModelLoader loads a model with weights in npz file.

Parameters

- **npz_file** (*str*) – name of npz_file
- **weights** – if provided, only load names in weights from npz file
- **variable_scope** – if provided, load all weights in this scope, otherwise load from default variable scope.

Examples:

```
# TODO
```

5.2.6 ModelSync

class neuralgym.callbacks.**ModelSync**(*pstep*, *from_namescope*, *to_namescope*, *step_start=False*)

ModelSync.

Currently it only supports sync trainable variables from one namescope to another namescope, which is enough for reinforcement learning.

Parameters

- **pstep** (*int*) – Sync every pstep.
- **from_namescope** (*str*) – Sync from from_namescope.
- **to_namescope** (*str*) – Sync to to_namescope.
- **step_start** – Sync at step_start, otherwise at step_end.

Examples:

```
# TODO
```

run(*sess*, *step*)

Run model sync

5.2.7 SummaryWriter

class neuralgym.callbacks.**SummaryWriter**(*pstep*, *summary_writer*, *summary*)

Periodically add summary.

Parameters

- **pstep** (*int*) – Call summary writer every pstep.
- **summary_writer** – Tensorflow summary writer.
- **summary** – Tensorflow summary collection.

5.2.8 SecondaryTrainer

class neuralgym.callbacks.**SecondaryTrainer**(*pstep*, ***context*)
This callback preiodically train discriminator for generative adversarial networks. Note that with this callback, the training of GAN is alternatively between training generator and discriminator.

5.2.9 SecondaryMultiGPUTrainer

class neuralgym.callbacks.**SecondaryMultiGPUTrainer**(*pstep*, ***context*)
SecondaryMultiGPUTrainer.

CHAPTER 6

neuralgym.ops

6.1 layers

layers

neuralgym.ops.layers.**HWNC_to_NHWC**(*x*, *name='HWNC_to_NHWC'*)
Convert data format from HWNC to NHWC, may be used for re-indexing.

neuralgym.ops.layers.**NCHW_to_NHWC**(*x*, *name='NCHW_to_NHWC'*)
Convert data format from NCHW to NHWC.

neuralgym.ops.layers.**NHWC_to_HWNC**(*x*, *name='NHWC_to_HWNC'*)
Convert data format from NHWC to HWNC, may be used for re-indexing.

neuralgym.ops.layers.**NHWC_to_NCHW**(*x*, *name='NHWC_to_NCHW'*)
Convert data format from NHWC to NCHW.

neuralgym.ops.layers.**apply_activation**(*x*, *relu*, *activation_fn*, *name='activation'*)
Wrapper for apply activation.

Note activation_fn has higher execution level.

neuralgym.ops.layers.**avg_pool**(*x*, *ksize=2*, *stride=2*, *padding='SAME'*, *name='avg_pool'*)
Average pooling wrapper.

neuralgym.ops.layers.**batch_transformer**(*U*, *thetas*, *out_size*, *name='BatchSpatialTransformer'*)
Batch Spatial Transformer Layer

Parameters

- **U** (*float*) –
- **of inputs [num_batch,height,width,num_channels]** (*tensor*) –
- **thetas** (*float*) –
- **set of transformations for each input [num_batch, num_transforms,6]** (*a*) –

- **out_size** (*int*) –
- **size of the output [out_height,out_width]** (*the*) –

Returns: float Tensor of size [num_batch*num_transforms,out_height,out_width,num_channels]

neuralgym.ops.layers.**bilinear_upsample**(*x*, *scale=2*)
Bilinear upsample.

Caffe bilinear upsample forked from https://github.com/ppwwyyxx/tensorpack Deterministic bilinearly-upsample the input images.

Parameters

- **x** (*tf.Tensor*) – a NHWC tensor
- **scale** (*int*) – the upsample factor

Returns a NHWC tensor.

Return type tf.Tensor

neuralgym.ops.layers.**concatenated_relu**(*x*, *name='concatenated_relu'*)
Concatenated relu wrapper.

neuralgym.ops.layers.**flatten**(*x*, *name='flatten'*)
Flatten wrapper.

neuralgym.ops.layers.**get_variable**(*name*, *shape*, *initializer*, *weight_decay=0.0*, *dtype='float'*, *trainable=True*, *freeze_weights=False*)
Simple wrapper for get_variable.

neuralgym.ops.layers.**max_pool**(*x*, *ksize=2*, *stride=2*, *padding='SAME'*, *name='max_pool'*)
Max pooling wrapper.

neuralgym.ops.layers.**moving_average_var**(*x*, *decay=0.99*, *initial_value=0.0*, *name='moving_average_var'*)
Moving_average_var.

neuralgym.ops.layers.**pixel_flow**(*x*, *offset*, *interpolation='bilinear'*, *name='pixel_flow'*)
pixel_flow: an operation to reorder pixels according to offsets.

Parameters

- **x** (*tf.Tensor*) – NHWC
- **offset** (*tf.Tensor*) – NHW2, 2 indicates (h, w) coordinates offset
- **interpolation** – bilinear, softmax
- **name** – name of module

References

[1] Spatial Transformer Networks: https://arxiv.org/abs/1506.02025 [2] https://github.com/ppwwyyxx/tensorpack

neuralgym.ops.layers.**scaled_elu**(*x*, *name='scaled_elu'*)
Scaled elu wrapper.

neuralgym.ops.layers.**transformer**(*U*, *theta*, *out_size=None*, *name='SpatialTransformer'*)
Spatial Transformer Layer.

Forked from tensorflow/models transformer.

6.2 summary_ops

summary ops.

neuralgym.ops.summary_ops.**scalar_summary**(*name*, *value*, *sess=None*, *summary_writer=None*, *step=None*)

Add scalar summary.

In addition to summary tf.Tensor and tf.Variable, this function supports summary of constant values by creating placeholder.

Example usage:

```
>>> scalar_summary('lr', lr)
```

Parameters

- **name** – name of summary variable
- **value** – numpy or tensorflow tensor
- **summary_writer** – if summary writer is provided, write to summary instantly
- **step** – if summary writer is provided, write to summary with step

Returns None

neuralgym.ops.summary_ops.**filters_summary**(*kernel*, *rescale=True*, *name='kernel'*)

Visualize filters and write to image summary.

Parameters

- **kernel** – kernel tensor
- **rescale** – rescale weights to [0, 1]

Returns None

neuralgym.ops.summary_ops.**images_summary**(*images*, *name*, *max_outs*, *color_format='BGR'*)

Summary images.

Note that images should be scaled to [-1, 1] for 'RGB' or 'BGR', [0, 1] for 'GREY'.

Parameters

- **images** – images tensor (in NHWC format)
- **name** – name of images summary
- **max_outs** – max_outputs for images summary
- **color_format** – 'BGR', 'RGB' or 'GREY'

Returns None

neuralgym.ops.summary_ops.**gradients_summary**(*y*, *x*, *norm=<function abs>*, *name='gradients_y_wrt_x'*)

Summary gradients w.r.t. x.

Sum of norm of $\nabla_x y$.

Parameters

- **y** – y
- **x** – w.r.t x

- **norm** – norm function, default is tf.abs
- **name** – name of gradients summary

Returns None

6.3 loss_ops

loss related functions

neuralgym.ops.loss_ops.**huber_loss**(*x*, *delta=1.0*, *name='huber_loss'*)
Huber loss: https://en.wikipedia.org/wiki/Huber_loss.

Deprecated. Please use tensorflow huber loss implementation.

neuralgym.ops.loss_ops.**l1_loss**(*x*, *y*, *name='l1_loss'*)
L1 loss: mean(abs(x-y)).

neuralgym.ops.loss_ops.**l2_loss**(*x*, *y*, *name='l2_loss'*)
L2_loss: mean((x-y) ** 2).

neuralgym.ops.loss_ops.**tv_loss**(*x*, *name='tv_loss'*)
tv_loss.

Deprecated. Please use tensorflow total_variation loss implementation.

6.4 image_ops

image related ops.

neuralgym.ops.image_ops.**np_random_crop**(*image*, *shape*, *align=True*)
Random crop.

shape from image.

Parameters

- **image** – numpy image, 2d or 3d
- **shape** – (height, width)

Returns numpy image

neuralgym.ops.image_ops.**np_scale_to_shape**(*image*, *shape*, *align=True*)
Scale the image.

The minimum side of height or width will be scaled to or larger than shape.

Parameters

- **image** – numpy image, 2d or 3d
- **shape** – (height, width)

Returns numpy image

CHAPTER 7

neuralgym.train

class neuralgym.train.**Trainer**(*primary=True*, ***context*)

Bases: object

Trainer class for train iterative algorithm on single GPU.

There are two types of trainer in neuralgym: primary trainer and secondary trainer. For primary trainer, tensorflow related instances and configurations will be initialized, e.g. init all variables, summary writer, session, start_queue_runner and others. For the secondary trainer only train_ops and losses are iteratively updated/ran.

add_callbacks(*callbacks*)

Add callbacks.

Parameters **callbacks** – dict of callbacks

init_primary_trainer()

Initialize primary trainer context including:

- log_dir
- global_step
- sess_config
- allow_growth
- summary writer
- saver
- global_variables_initializer
- start_queue_runners

progress_logger(*step*, *loss*)

Progress bar for logging.

Note all statistics are averaged over epoch.

train()

Start training with callbacks.

class neuralgym.train.**MultiGPUTrainer**(**context*)

Bases: neuralgym.train.trainer.Trainer

Trainer class for train iterative algorithm on multi GPUs.

Parameters

- **num_gpus** (*int*) – Number of GPU(s) for training.
- **async_train** (*bool*) – Asynchronous train or not.

train()

Start training with callbacks.

CHAPTER 8

neuralgym.utils

neuralgym.utils.**callback_log**(*texts*)
Callback_log will show caller's location.

Parameters **texts** (*str*) – Text to show.

neuralgym.utils.**warning_log**(*texts*)
Warning_log will show caller's location and red texts.

Parameters **texts** (*str*) – Text to show.

neuralgym.utils.**error_log**(*texts*)
Error_log will show caller's location, red texts and raise RuntimeError.

Parameters **texts** (*str*) – Text to show.

neuralgym.utils.**colored_log**(*prompt*, *texts*, *color='green'*, *bold=True*, *highlight=False*)
Show colored logs.

neuralgym.utils.**get_sess**(*sess=None*)
Get default session if sess is None.

Parameters **sess** – Valid sess or None.

Returns Valid sess or get default sess.

class neuralgym.utils.**ProgressBar**
Bases: object

Visualize progress.

It displays a progress bar in console with time recorder and statistics.

progress(*progress*, *texts=''*)
Update progress bar with current progress and additional texts.

Parameters

- **progress** (*float*) – A float between [0,1] indicating progress.
- **texts** (*str*) – additional texts (e.g. statistics) appear at the end of progress bar.

restart()
Restart time recorder and progress recorder.

CHAPTER 9

neuralgym.data

class neuralgym.data.**Dataset**

Bases: object

Base class for datasets.

Dataset members are automatically logged except members with name ending of '_', e.g. '**self.fnamelists_**'.

data_pipeline(*batch_size*)

Return batch data with batch size, e.g. return batch_image or return (batch_data, batch_label).

Parameters **batch_size** (*int*) – Batch size.

maybe_download_and_extract()

Abstract class: dataset maybe need download items.

view_dataset_info()

Function to view current dataset information.

class neuralgym.data.**DataFromFNames**(*fnamelists*, *shapes*, *random=False*, *random_crop=False*, *fn_preprocess=None*, *dtypes=tf.float32*, *enqueue_size=32*, *queue_size=256*, *nthreads=16*, *return_fnames=False*, *filetype='image'*)

Bases: neuralgym.data.dataset.Dataset

Data pipeline from list of filenames.

Parameters

- **fnamelists** (*list*) – A list of filenames or tuple of filenames, e.g. ['image_001.png', ...] or [('pair_image_001_0.png', 'pair_image_001_1.png'), ...].
- **shapes** (*tuple*) – Shapes of data, e.g. [256, 256, 3] or [[256, 256, 3], [1]].
- **random** (*bool*) – Read from *fnamelists* randomly (default to False).
- **random_crop** (*bool*) – If random crop to the shape from raw image or directly resize raw images to the shape.
- **dtypes** (*tf.Type*) – Data types, default to tf.float32.

- **enqueue_size** (*int*) – Enqueue size for pipeline.
- **enqueue_size** – Enqueue size for pipeline.
- **nthreads** (*int*) – Parallel threads for reading from data.
- **return_fnames** (*bool*) – If True, data_pipeline will also return fnames (last tensor).
- **filetype** (*str*) – Currently only support image.

Examples

```
>>> fnames = ['img001.png', 'img002.png', ..., 'img999.png']
>>> data = ng.data.DataFromFNames(fnames, [256, 256, 3])
>>> images = data.data_pipeline(128)
>>> sess = tf.Session(config=tf.ConfigProto())
>>> tf.train.start_queue_runners(sess)
>>> for i in range(5): sess.run(images)
```

To get file lists, you can either use file:

```
with open('data/images.flist') as f:
    fnames = f.read().splitlines()
```

or glob:

```
import glob
fnames = glob.glob('data/*.png')
```

data_pipeline(*batch_size*)

Batch data pipeline.

Parameters **batch_size** (*int*) – Batch size.

Returns

A tensor with shape [batch_size] and self.shapes e.g. if self.shapes = ([256, 256, 3], [1]), then return [[batch_size, 256, 256, 3], [batch_size, 1]].

Python Module Index

n

Index

O

P

R

S

T

U

V

W