
tensorcv documentation

Release 0.1

Qian Ge

Sep 12, 2018

Contents

1	tensorcv	1
	Python Module Index	21

1.1 tensorcv package

1.1.1 Subpackages

tensorcv.callbacks package

Submodules

tensorcv.callbacks.base module

```
class tensorcv.callbacks.base.Callback
    Bases: object
    base class for callbacks
    after_epoch()
    after_run(ret, val)
    after_train()
    before_epoch()
    before_inference()
    before_run(ret)
    before_train()
    epochs_completed
    global_step
    setup_graph(trainer)
    trigger()
```

`trigger_epoch()`

`trigger_step()`

class `tensorcv.callbacks.base.ProxyCallback` (*cb*)

Bases: `tensorcv.callbacks.base.Callback`

tensorcv.callbacks.debug module

class `tensorcv.callbacks.debug.CheckScalar` (*tensors, periodic=1*)

Bases: `tensorcv.callbacks.base.Callback`

print scalar tensor values during training .. attribute:: `_tensors`

`_names`

`__init__` (*tensors, periodic=1*)

init CheckScalar object :param *tensors*: list[string] A tensor name or list of tensor names

tensorcv.callbacks.group module

class `tensorcv.callbacks.group.Callbacks` (*cbs*)

Bases: `tensorcv.callbacks.base.Callback`

group all the callback

`get_hooks()`

tensorcv.callbacks.hooks module

class `tensorcv.callbacks.hooks.Callback2Hook` (*cb*)

Bases: `tensorflow.python.training.session_run_hook.SessionRunHook`

`after_run` (*rct, val*)

Called after each call to `run()`.

The *run_values* argument contains results of requested ops/tensors by *before_run()*.

The *run_context* argument is the same one send to *before_run* call. *run_context.request_stop()* can be called to stop the iteration.

If *session.run()* raises any exceptions then *after_run()* is not called.

Parameters

- **run_context** – A *SessionRunContext* object.
- **run_values** – A *SessionRunValues* object.

`before_run` (*rct*)

Called before each call to `run()`.

You can return from this call a *SessionRunArgs* object indicating ops or tensors to add to the upcoming *run()* call. These ops/tensors will be run together with the ops/tensors originally passed to the original *run()* call. The run args you return can also contain feeds to be added to the *run()* call.

The *run_context* argument is a *SessionRunContext* that provides information about the upcoming *run()* call: the originally requested op/tensors, the TensorFlow Session.

At this point graph is finalized and you can not add ops.

Parameters `run_context` – A *SessionRunContext* object.

Returns None or a *SessionRunArgs* object.

class `tensorcv.callbacks.hooks.Infer2Hook` (*inferencer*)

Bases: `tensorflow.python.training.session_run_hook.SessionRunHook`

after_run (*rct, val*)

Called after each call to `run()`.

The `run_values` argument contains results of requested ops/tensors by `before_run()`.

The `run_context` argument is the same one send to `before_run` call. `run_context.request_stop()` can be called to stop the iteration.

If `session.run()` raises any exceptions then `after_run()` is not called.

Parameters

- `run_context` – A *SessionRunContext* object.
- `run_values` – A *SessionRunValues* object.

before_run (*rct*)

Called before each call to `run()`.

You can return from this call a *SessionRunArgs* object indicating ops or tensors to add to the upcoming `run()` call. These ops/tensors will be run together with the ops/tensors originally passed to the original `run()` call. The run args you return can also contain feeds to be added to the `run()` call.

The `run_context` argument is a *SessionRunContext* that provides information about the upcoming `run()` call: the originally requested op/tensors, the TensorFlow Session.

At this point graph is finalized and you can not add ops.

Parameters `run_context` – A *SessionRunContext* object.

Returns None or a *SessionRunArgs* object.

class `tensorcv.callbacks.hooks.Prediction2Hook` (*prediction*)

Bases: `tensorflow.python.training.session_run_hook.SessionRunHook`

after_run (*rct, val*)

Called after each call to `run()`.

The `run_values` argument contains results of requested ops/tensors by `before_run()`.

The `run_context` argument is the same one send to `before_run` call. `run_context.request_stop()` can be called to stop the iteration.

If `session.run()` raises any exceptions then `after_run()` is not called.

Parameters

- `run_context` – A *SessionRunContext* object.
- `run_values` – A *SessionRunValues* object.

before_run (*rct*)

Called before each call to `run()`.

You can return from this call a *SessionRunArgs* object indicating ops or tensors to add to the upcoming `run()` call. These ops/tensors will be run together with the ops/tensors originally passed to the original `run()` call. The run args you return can also contain feeds to be added to the `run()` call.

The `run_context` argument is a *SessionRunContext* that provides information about the upcoming `run()` call: the originally requested op/tensors, the TensorFlow Session.

At this point graph is finalized and you can not add ops.

Parameters `run_context` – A *SessionRunContext* object.

Returns None or a *SessionRunArgs* object.

tensorcv.callbacks.inference module

```
class tensorcv.callbacks.inference.FeedInference (inputs,          periodic=1,    infer-
                                                encers=[],    extra_cbs=None,    infer_batch_size=None)
    Bases: tensorcv.callbacks.inference.InferenceBase
    default inferencer: inference_list = InferImages('generator/gen_image', prefix = 'gen')
class tensorcv.callbacks.inference.GANInference (inputs=None,    periodic=1,    infer-
                                                encers=None, extra_cbs=None)
    Bases: tensorcv.callbacks.inference.InferenceBase
class tensorcv.callbacks.inference.FeedInferenceBatch (inputs,          periodic=1,
                                                         batch_count=10,    infer-
                                                         encers=[],    extra_cbs=None,
                                                         infer_batch_size=None)
    Bases: tensorcv.callbacks.inference.FeedInference
    do not use all validation data
```

tensorcv.callbacks.inferencer module

```
class tensorcv.callbacks.inferencer.InferencerBase
    Bases: tensorcv.callbacks.base.Callback
    after_inference ()
    before_inference ()
        process before every inference
    get_fetch (val)
    put_fetch ()
    setup_inferencer ()
class tensorcv.callbacks.inferencer.InferImages (im_name, prefix=None, color=False,
                                                         tanh=False)
    Bases: tensorcv.callbacks.inferencer.InferencerBase
class tensorcv.callbacks.inferencer.InferScalars (scaler_names,          sum-
                                                         mary_names=None)
    Bases: tensorcv.callbacks.inferencer.InferencerBase
class tensorcv.callbacks.inferencer.InferOverlay (im_name, prefix=None, color=False,
                                                         tanh=False)
    Bases: tensorcv.callbacks.inferencer.InferImages
class tensorcv.callbacks.inferencer.InferMat (infer_save_name,    mat_name,    pre-
                                                         fix=None)
    Bases: tensorcv.callbacks.inferencer.InferImages
```


tensorcv.callbacks.inputs module

class tensorcv.callbacks.inputs.**FeedInput** (*dataflow, placeholders*)
 Bases: *tensorcv.callbacks.base.Callback*
 input using feed

tensorcv.callbacks.monitors module

class tensorcv.callbacks.monitors.**TrainingMonitor**
 Bases: *tensorcv.callbacks.base.Callback*
process_summary (*summary*)

class tensorcv.callbacks.monitors.**Monitors** (*mons*)
 Bases: *tensorcv.callbacks.monitors.TrainingMonitor*
 group monitors

class tensorcv.callbacks.monitors.**TFSummaryWriter**
 Bases: *tensorcv.callbacks.monitors.TrainingMonitor*
process_summary (*summary*)

tensorcv.callbacks.saver module

class tensorcv.callbacks.saver.**ModelSaver** (*max_to_keep=5,*
keep_checkpoint_every_n_hours=0.5,
periodic=1, checkpoint_dir=None,
var_collections='variables')
 Bases: *tensorcv.callbacks.base.Callback*

tensorcv.callbacks.summary module

class tensorcv.callbacks.summary.**TrainSummary** (*key=None, periodic=1*)
 Bases: *tensorcv.callbacks.base.Callback*

tensorcv.callbacks.trigger module

class tensorcv.callbacks.trigger.**PeriodicTrigger** (*trigger_cb, every_k_steps=None, every_k_epochs=None*)
 Bases: *tensorcv.callbacks.base.ProxyCallback*
 may not need

Module contents

tensorcv.dataflow package

Subpackages

tensorcv.dataflow.dataset package

Submodules

tensorcv.dataflow.dataset.BSDS500 module

```
class tensorcv.dataflow.dataset.BSDS500.BSDS500 (name, data_dir="", shuffle=True, normalize=None, is_mask=False, normalize_fnc=<function identity>, resize=None)

    Bases: tensorcv.dataflow.image.ImageFromFile
```

```
class tensorcv.dataflow.dataset.BSDS500.BSDS500HED (name, data_dir="", shuffle=True, normalize=None, is_mask=False, normalize_fnc=<function identity>, resize=None)

    Bases: tensorcv.dataflow.dataset.BSDS500.BSDS500
```

tensorcv.dataflow.dataset.CIFAR module

```
class tensorcv.dataflow.dataset.CIFAR.CIFAR (data_dir="", shuffle=True, normalize=None)
    Bases: tensorcv.dataflow.base.RNGDataFlow

    next_batch()

    size()
```

tensorcv.dataflow.dataset.MNIST module

```
class tensorcv.dataflow.dataset.MNIST.MNIST (name, data_dir="", shuffle=True, normalize=None)
    Bases: tensorcv.dataflow.base.RNGDataFlow

    next_batch()

    size()

class tensorcv.dataflow.dataset.MNIST.MNISTLabel (name, data_dir="", shuffle=True, normalize=None)
    Bases: tensorcv.dataflow.dataset.MNIST.MNIST

    next_batch()
```

Module contents

Submodules

tensorcv.dataflow.base module

```
class tensorcv.dataflow.base.DataFlow
    Bases: object
    base class for dataflow

    after_reading()

    before_read_setup(**kwargs)

    epochs_completed

    next_batch()

    next_batch_dict()

    reset_epochs_completed(val)

    reset_state()

    set_batch_size(batch_size)

    setup(epoch_val, batch_size, **kwargs)

    size()

class tensorcv.dataflow.base.RNGDataFlow
    Bases: tensorcv.dataflow.base.DataFlow

    suffle_data()
```

tensorcv.dataflow.common module

```
tensorcv.dataflow.common.dense_to_one_hot(labels_dense, num_classes)
    Convert class labels from scalars to one-hot vectors.

tensorcv.dataflow.common.get_file_list(file_dir, file_ext, sub_name=None)

tensorcv.dataflow.common.get_folder_list(folder_dir)

tensorcv.dataflow.common.get_folder_names(folder_dir)

tensorcv.dataflow.common.input_val_range(in_mat)

tensorcv.dataflow.common.load_image(im_path, read_channel=None, pf=<function identity>,
    resize=None, resize_crop=None)

tensorcv.dataflow.common.print_warning(warning_str)

tensorcv.dataflow.common.reverse_label_dict(label_dict)

tensorcv.dataflow.common.tanh_normalization(data, half_in_val)
```

tensorcv.dataflow.image module

```
class tensorcv.dataflow.image.ImageData (ext_name, data_dir=", shuffle=True, normalize=None)
```

Bases: *tensorcv.dataflow.base.RNGDataFlow*

next_batch()

size()

```
class tensorcv.dataflow.image.DataFromFile (ext_name, data_dir=", num_channel=None,
                                             shuffle=True, normalize=None,
                                             batch_dict_name=None, normalize_fnc=<function identity>)
```

Bases: *tensorcv.dataflow.base.RNGDataFlow*

Base class for image from files

get_sample_data()

next_batch()

next_batch_dict()

```
class tensorcv.dataflow.image.ImageLabelFromFolder (ext_name, data_dir=",
                                                    num_channel=None, label_dict=None, num_class=None,
                                                    one_hot=False, shuffle=True, normalize=None,
                                                    resize=None, resize_crop=None,
                                                    batch_dict_name=None, pf=<function identity>)
```

Bases: *tensorcv.dataflow.image.ImageFromFile*

read image data with label in subfolder name

```
__init__ (ext_name, data_dir=", num_channel=None, label_dict=None, num_class=None,
          one_hot=False, shuffle=True, normalize=None, resize=None, resize_crop=None,
          batch_dict_name=None, pf=<function identity>)
```

Parameters **label_dict** (*dict*) – empty or full

get_data_list()

get_label_list()

set_data_list (*new_data_list*)

size()

```
class tensorcv.dataflow.image.ImageLabelFromFile (ext_name, data_dir=",
                                                    label_file_name=",
                                                    num_channel=None, one_hot=False,
                                                    label_dict={}, num_class=None,
                                                    shuffle=True, normalize=None,
                                                    resize=None, resize_crop=None,
                                                    batch_dict_name=None,
                                                    pf=<function identity>)
```

Bases: *tensorcv.dataflow.image.ImageLabelFromFolder*

read image data with label in a separate file txt

```
class tensorcv.dataflow.image.ImageFromFile (ext_name, data_dir="", num_channel=None,  
                                              shuffle=True, normalize=None, normal-  
                                              ize_fnc=<function identity>, resize=None,  
                                              resize_crop=None, batch_dict_name=None,  
                                              pf=<function identity>)
```

Bases: *tensorcv.dataflow.image.DataFromFile*

get_data_list ()

set_data_list (*new_data_list*)

set_pf (*pf*)

size ()

suffle_data ()

```
class tensorcv.dataflow.image.ImageDenseLabel (ext_name, im_pre, label_pre,  
                                              mask_pre=None, data_dir="",  
                                              num_channel=None, shuffle  
                                              =True, normalize=None, nor-  
                                              malize_fnc=<function identity>,  
                                              resize=None, resize_crop=None,  
                                              batch_dict_name=None,  
                                              is_binary=False)
```

Bases: *tensorcv.dataflow.image.ImageFromFile*

get_data_list ()

get_label_list ()

set_data_list (*new_data_list*)

tensorcv.dataflow.matlab module

```
class tensorcv.dataflow.matlab.MatlabData (data_dir="", mat_name_list=None,  
                                              mat_type_list=None, shuffle=True, normal-  
                                              ize=None)
```

Bases: *tensorcv.dataflow.base.RNGDataFlow*

dataflow from .mat file with mask

next_batch ()

size ()

tensorcv.dataflow.randoms module

```
class tensorcv.dataflow.randoms.RandomVec (len_vec=100)
```

Bases: *tensorcv.dataflow.base.DataFlow*

random vector input

next_batch ()

reset_state ()

size ()

Module contents

tensorcv.models package

Submodules

tensorcv.models.base module

```
class tensorcv.models.base.ModelDes
    Bases: object
    base model for ModelDes

    create_graph()
    create_model(inputs=None)
    ex_init_model(dataflow, trainer)
    get_batch_size()
    get_global_step
    get_graph_feed()
    get_prediction_placeholder()
    get_train_placeholder()
    model_input
    set_batch_size(val)
    set_dropout(dropout_placeholder, keep_prob=0.5)
    set_is_training(is_training=True)
    set_model_input(inputs=None)
    set_prediction_placeholder(plhs=None)
    set_train_placeholder(plhs=None)
    setup_summary()

class tensorcv.models.base.BaseModel
    Bases: tensorcv.models.base.ModelDes
    Model with single loss and single optimizer

    default_collection
    get_grads()
    get_loss()
    get_optimizer()

class tensorcv.models.base.GANBaseModel(input_vec_length, learning_rate)
    Bases: tensorcv.models.base.ModelDes
    Base model for GANs

    d_collection
```

```
def_loss (dis_loss_fnc, gen_loss_fnc)
    update definition of loss functions

g_collection

get_discriminator_grads ()

get_discriminator_loss ()

get_discriminator_optimizer ()

get_gen_data ()

get_generator_grads ()

get_generator_loss ()

get_generator_optimizer ()

get_graph_feed ()

get_random_vec_placeholder ()

get_sample_gen_data ()
```

tensorcv.models.layers module

tensorcv.models.layers.**batch_flatten**(x)
Flatten the tensor except the first dimension.

tensorcv.models.layers.**batch_norm**(x, train=True, name='bn')
batch normal

Parameters

- **x** (*tf.tensor*) – a tensor
- **name** (*str*) – name scope
- **train** (*bool*) – whether training or not

Returns tf.tensor with name 'name'

tensorcv.models.layers.**conv**(x, filter_size, out_dim, name='conv', stride=1, padding='SAME',
nl=<function identity>, data_dict=None, init_w=None,
init_b=None, use_bias=True, wd=None, trainable=True)

2D convolution

Parameters

- **x** (*tf.tensor*) – a 4D tensor Input number of channels has to be known
- **filter_size** (*int* or *list* with length 2) – size of filter
- **out_dim** (*int*) – number of output channels
- **name** (*str*) – name scope of the layer
- **stride** (*int* or *list*) – stride of filter
- **padding** (*str*) – 'VALID' or 'SAME'
- **init_b** (*init_w*,) – initializer for weight and bias variables. Default to 'random_normal_initializer'
- **nl** – a function

Returns tf.tensor with name 'output'

```
tensorcv.models.layers.dconv(x, filter_size, out_dim=None, out_shape=None,
                             out_shape_by_tensor=None, name='dconv', stride=2,
                             padding='SAME', nl=<function identity>, data_dict=None,
                             init_w=None, init_b=None, wd=None, trainable=True)
```

2D deconvolution

Parameters

- **x** (*tf.tensor*) – a 4D tensor Input number of channels has to be known
- **filter_size** (*int* or *list* with length 2) – size of filter
- **out_dim** (*int*) – number of output channels
- **out_shape** (*list(int)*) – shape of output without None
- **out_shape_by_tensor** (*tf.tensor*) – a tensor has the same shape of output except the out_dim
- **name** (*str*) – name scope of the layer
- **stride** (*int* or *list*) – stride of filter
- **padding** (*str*) – 'VALID' or 'SAME'
- **init** – initializer for variables. Default to 'random_normal_initializer'
- **nl** – a function

Returns tf.tensor with name 'output'

```
tensorcv.models.layers.dropout(x, keep_prob, is_training, name='dropout')
```

Dropout

Parameters

- **x** (*tf.tensor*) – a tensor
- **keep_prob** (*float*) – keep prbability of dropout
- **is_training** (*bool*) – whether training or not
- **name** (*str*) – name scope

Returns tf.tensor with name 'name'

```
tensorcv.models.layers.fc(x, out_dim, name='fc', nl=<function identity>, init_w=None,
                           init_b=None, data_dict=None, wd=None, trainable=True,
                           re_dict=False)
```

Fully connected layer

Parameters

- **x** (*tf.tensor*) – a tensor to be flattened The first dimension is the batch dimension
- **num_out** (*int*) – dimension of output
- **name** (*str*) – name scope of the layer
- **init** – initializer for variables. Default to 'random_normal_initializer'
- **nl** – a function

Returns tf.tensor with name 'output'

```
tensorcv.models.layers.get_shape2D(in_val)
```

Return a 2D shape

Parameters `in_val` (*int* or *list* with length 2)–

Returns list with length 2

`tensorcv.models.layers.get_shape4D(in_val)`

Return a 4D shape

Parameters `in_val` (*int* or *list* with length 2)–

Returns list with length 4

`tensorcv.models.layers.global_avg_pool(x, name='global_avg_pool', data_format='NHWC')`

`tensorcv.models.layers.leaky_relu(x, leak=0.2, name='LeakyRelu')`

Allow a small non-zero gradient when the unit is not active

Parameters

- `x` (*tf.tensor*) – a tensor
- `leak` (*float*) – Default to 0.2

Returns *tf.tensor* with name 'name'

`tensorcv.models.layers.max_pool(x, name='max_pool', filter_size=2, stride=None, padding='VALID')`

Max pooling layer

Parameters

- `x` (*tf.tensor*) – a tensor
- `name` (*str*) – name scope of the layer
- `filter_size` (*int* or *list* with length 2) – size of filter
- `stride` (*int* or *list* with length 2) – Default to be the same as shape
- `padding` (*str*) – 'VALID' or 'SAME'. Use 'SAME' for FCN.

Returns *tf.tensor* with name 'name'

`tensorcv.models.layers.new_biases(name, idx, shape, initializer=None, data_dict=None, trainable=True)`

`tensorcv.models.layers.new_normal_variable(name, shape=None, trainable=True, std-dev=0.002)`

`tensorcv.models.layers.new_variable(name, idx, shape, initializer=None)`

`tensorcv.models.layers.new_weights(name, idx, shape, initializer=None, wd=None, data_dict=None, trainable=True)`

tensorcv.models.losses module

`tensorcv.models.losses.GAN_discriminator_loss(d_real, d_fake, name='d_loss')`

`tensorcv.models.losses.GAN_generator_loss(d_fake, name='g_loss')`

`tensorcv.models.losses.comp_loss_fake(discrim_output)`

`tensorcv.models.losses.comp_loss_real(discrim_output)`

Module contents

tensorcv.predicts package

Submodules

tensorcv.predicts.base module

```
class tensorcv.predicts.base.Predictor (config)
    Bases: object

    Base class for a predictor. Used to run all predictions.

    config
        PredictConfig – the config used for this predictor

    model
        ModelDes

    input
        DataFlow

    sess
        tf.Session

    hooked_sess
        tf.train.MonitoredSession

    __init__ (config)
        Inits Predictor with config (PredictConfig).

        Will create session as well as monitored sessions for each predictions, and load pre-trained parameters.

        Parameters config (PredictConfig) – the config used for this predictor

    after_prediction ()

    run_predict ()
        Run predictions and the process after finishing predictions.
```

tensorcv.predicts.config module

```
class tensorcv.predicts.config.PredictConfig (dataflow=None, model=None,
                                              model_dir=None, model_name="", re-
                                              store_vars=None, session_creator=None,
                                              predictions=None, batch_size=1, de-
                                              fault_dirs=None)

    Bases: object

    __init__ (dataflow=None, model=None, model_dir=None, model_name="", restore_vars=None, ses-
              session_creator=None, predictions=None, batch_size=1, default_dirs=None)
        Args:

    callbacks
```

tensorcv.predicts.predictions module

class tensorcv.predicts.predictions.**PredictionImage** (*prediction_image_tensors*,
save_prefix, *merge_im=False*,
tanh=False, *color=False*)

Bases: tensorcv.predicts.predictions.PredictionBase

Predict image output and save as files.

Images are saved every batch. Each batch result can be save in one image or individule images.

__init__ (*prediction_image_tensors*, *save_prefix*, *merge_im=False*, *tanh=False*, *color=False*)

Parameters

- **prediction_image_tensors** (*list*) – a list of tensor names
- **save_prefix** (*list*) – a list of file prefix for saving each tensor in prediction_image_tensors
- **merge_im** (*bool*) – merge output of one batch or not

class tensorcv.predicts.predictions.**PredictionScalar** (*prediction_scalar_tensors*,
print_prefix)

Bases: tensorcv.predicts.predictions.PredictionBase

__init__ (*prediction_scalar_tensors*, *print_prefix*)

Parameters

- **prediction_scalar_tensors** (*list*) – a list of tensor names
- **print_prefix** (*list*) – a list of name prefix for printing each tensor in prediction_scalar_tensors

class tensorcv.predicts.predictions.**PredictionMat** (*prediction_tensors*, *save_prefix*)

Bases: tensorcv.predicts.predictions.PredictionBase

class tensorcv.predicts.predictions.**PredictionMeanScalar** (*prediction_scalar_tensors*,
print_prefix)

Bases: *tensorcv.predicts.predictions.PredictionScalar*

class tensorcv.predicts.predictions.**PredictionOverlay** (*prediction_image_tensors*,
save_prefix, *merge_im=False*,
tanh=False, *color=False*)

Bases: *tensorcv.predicts.predictions.PredictionImage*

tensorcv.predicts.simple module

class tensorcv.predicts.simple.**SimpleFeedPredictor** (*config*)

Bases: *tensorcv.predicts.base.Predictor*

predictor with feed input

Module contents

tensorcv.train package

Submodules

tensorcv.train.base module

```
class tensorcv.train.base.Trainer (config)
    Bases: object
    base class for trainer

    epochs_completed

    get_global_step

    main_loop()

    register_callback(cb)

    register_monitor(monitor)

    setup()

    setup_graph()

    train()
```

tensorcv.train.config module

```
class tensorcv.train.config.TrainConfig (dataflow=None, model=None, callbacks=[],
                                         session_creator=None, monitors=None,
                                         batch_size=1, max_epoch=100, summary_periodic=None,
                                         is_load=False, model_name=None, default_dirs=None)

    Bases: object

    callbacks

class tensorcv.train.config.GANTrainConfig (dataflow=None, model=None, discriminator_callbacks=[],
                                              generator_callbacks=[],
                                              session_creator=None, monitors=None,
                                              batch_size=1, max_epoch=100,
                                              summary_d_periodic=None, summary_g_periodic=None,
                                              default_dirs=None)

    Bases: tensorcv.train.config.TrainConfig

    dis_callbacks

    gen_callbacks
```

tensorcv.train.simple module

```
class tensorcv.train.simple.SimpleFeedTrainer (config)
    Bases: tensorcv.train.base.Trainer
    single optimizer
```

Module contents

tensorcv.utils package

Submodules

tensorcv.utils.common module

`tensorcv.utils.common.apply_mask(input_matrix, mask)`

Get partition of input_matrix using index 1 in mask.

Parameters

- **input_matrix** (*Tensor*) – A Tensor
- **mask** (*int*) – A Tensor of type int32 with indices in {0, 1}. Shape has to be the same as input_matrix.

Returns A Tensor with elements from data with entries in mask equal to 1.

`tensorcv.utils.common.apply_mask_inverse(input_matrix, mask)`

Get partition of input_matrix using index 0 in mask.

Parameters

- **input_matrix** (*Tensor*) – A Tensor
- **mask** (*int*) – A Tensor of type int32 with indices in {0, 1}. Shape has to be the same as input_matrix.

Returns A Tensor with elements from data with entries in mask equal to 0.

`tensorcv.utils.common.get_tensors_by_names(names)`

Get a list of tensors by the input name list.

Parameters **names** (*str*) – A str or a list of str

Returns A list of tensors with name in input names.

Warning: If more than one tensor have the same name in the graph. This function will only return the tensor with name NAME:0.

`tensorcv.utils.common.deconv_size(input_height, input_width, stride=2)`

Compute the feature size (height and width) after filtering with a specific stride. Mostly used for setting the shape for deconvolution.

Parameters

- **input_height** (*int*) – height of input feature
- **input_width** (*int*) – width of input feature
- **stride** (*int*) – stride of the filter

Returns (*int, int*) – Height and width of feature after filtering.

`tensorcv.utils.common.match_tensor_save_name(tensor_names, save_names)`

Match tensor_names and corresponding save_names for saving the results of the tenors. If the number of tensors is less or equal to the length of save names, tensors will be saved using the corresponding names in save_names. Otherwise, tensors will be saved using their own names. Used for prediction or inference.

Parameters

- **tensor_names** (*str*) – List of tensor names
- **save_names** (*str*) – List of names for saving tensors

Returns (*list, list*) – List of tensor names and list of names to save the tensors.

tensorcv.utils.default module

tensorcv.utils.default.get_default_session_config(*memory_fraction=1*)

Default config of a TensorFlow session

Parameters **memory_fraction** (*float*) – Memory fraction of GPU for this session

Returns *tf.ConfigProto()* – Config of session.

tensorcv.utils.ssesscreate module

class tensorcv.utils.ssesscreate.NewSessionCreator(*target="", graph=None, config=None*)

Bases: tensorflow.python.training.monitored_session.SessionCreator

tf.train.SessionCreator for a new session

__init__ (*target="", graph=None, config=None*)

Init NewSessionCreator with target, graph and config.

Parameters

- **target** – same as *tf.Session.__init__()*.
- **graph** – same as *tf.Session.__init__()*.
- **config** – same as *tf.Session.__init__()*. Default to *utils.default.get_default_session_config()*.

create_session ()

Create session as well as initialize global and local variables

Returns A *tf.Session* object containing nodes for all of the operations in the underlying TensorFlow graph.

class tensorcv.utils.ssesscreate.ReuseSessionCreator(*sess*)

Bases: tensorflow.python.training.monitored_session.SessionCreator

tf.train.SessionCreator for reuse an existed session

__init__ (*sess*)

Init ReuseSessionCreator with an existed session.

Parameters **sess** (*tf.Session*) – an existed *tf.Session* object

create_session ()

Create session by reusing an existing session

Returns A reused *tf.Session* object containing nodes for all of the operations in the underlying TensorFlow graph.

tensorcv.utils.utils module

tensorcv.utils.utils.**get_rng**(*obj=None*)

This function is copied from [tensorpack](#). Get a good RNG seeded with time, pid and the object. :param obj: some object to use to generate random seed.

Returns *np.random.RandomState* – the RNG.

tensorcv.utils.viz module

tensorcv.utils.viz.**image_overlay**(*im_1, im_2, color=True, normalize=True*)

Overlay two images with the same size.

Parameters

- **im_1** (*np.ndarray*) – image array
- **im_2** (*np.ndarray*) – image array
- **color** (*bool*) – Whether convert intensity image to color image.
- **normalize** (*bool*) – If both color and normalize are True, will normalize the intensity so that it has minimum 0 and maximum 1.

Returns *np.ndarray* – an overlay image of $im_1 * 0.5 + im_2 * 0.5$

tensorcv.utils.viz.**intensity_to_rgb**(*intensity, cmap='jet', normalize=False*)

This function is copied from [tensorpack](#). Convert a 1-channel matrix of intensities to an RGB image employing a colormap. This function requires matplotlib. See [matplotlib colormaps](#) for a list of available colormap.

Parameters

- **intensity** (*np.ndarray*) – array of intensities such as saliency.
- **cmap** (*str*) – name of the colormap to use.
- **normalize** (*bool*) – if True, will normalize the intensity so that it has minimum 0 and maximum 1.

Returns *np.ndarray* – an RGB float32 image in range [0, 255], a colored heatmap.

tensorcv.utils.viz.**save_merge_images**(*images, merge_grid, save_path, color=False, tanh=False*)

Save multiple images with same size into one larger image.

The best size number is $\text{int}(\max(\sqrt{\text{image.shape}[0]}, \sqrt{\text{image.shape}[1]})) + 1$

Parameters

- **images** (*np.ndarray*) – A batch of image array to be merged with size [BATCH_SIZE, HEIGHT, WIDTH, CHANNEL].
- **merge_grid** (*list*) – List of length 2. The grid size for merge images.
- **save_path** (*str*) – Path for saving the merged image.
- **color** (*bool*) – Whether convert intensity image to color image.
- **tanh** (*bool*) – If True, will normalize the image in range [-1, 1] to [0, 1] (for GAN models).

Example

The batch_size is 64, then the size is recommended [8, 8]. The batch_size is 32, then the size is recommended [6, 6].

Module contents

1.1.2 Module contents

c

- `tensorcv.callbacks`, 6
- `tensorcv.callbacks.base`, 1
- `tensorcv.callbacks.debug`, 2
- `tensorcv.callbacks.group`, 2
- `tensorcv.callbacks.hooks`, 2
- `tensorcv.callbacks.inference`, 4
- `tensorcv.callbacks.inferencer`, 4
- `tensorcv.callbacks.inputs`, 5
- `tensorcv.callbacks.monitors`, 5
- `tensorcv.callbacks.saver`, 5
- `tensorcv.callbacks.summary`, 5
- `tensorcv.callbacks.trigger`, 5

d

- `tensorcv.dataflow`, 10
- `tensorcv.dataflow.base`, 7
- `tensorcv.dataflow.common`, 7
- `tensorcv.dataflow.dataset`, 7
- `tensorcv.dataflow.dataset.BSDS500`, 6
- `tensorcv.dataflow.dataset.CIFAR`, 6
- `tensorcv.dataflow.dataset.MNIST`, 6
- `tensorcv.dataflow.image`, 8
- `tensorcv.dataflow.matlab`, 9
- `tensorcv.dataflow.randoms`, 9

m

- `tensorcv.models`, 14
- `tensorcv.models.base`, 10
- `tensorcv.models.layers`, 11
- `tensorcv.models.losses`, 13

p

- `tensorcv.predicts`, 16
- `tensorcv.predicts.base`, 14
- `tensorcv.predicts.config`, 14
- `tensorcv.predicts.predictions`, 15
- `tensorcv.predicts.simple`, 15

t

- `tensorcv`, 20
- `tensorcv.train`, 17
- `tensorcv.train.base`, 16
- `tensorcv.train.config`, 16
- `tensorcv.train.simple`, 16

u

- `tensorcv.utils`, 20
- `tensorcv.utils.common`, 17
- `tensorcv.utils.default`, 18
- `tensorcv.utils.sesscreate`, 18
- `tensorcv.utils.utils`, 19
- `tensorcv.utils.viz`, 19

Symbols

__init__() (tensorcv.callbacks.debug.CheckScalar method), 2
 __init__() (tensorcv.dataflow.image.ImageLabelFromFolder method), 8
 __init__() (tensorcv.predicts.base.Predictor method), 14
 __init__() (tensorcv.predicts.config.PredictConfig method), 14
 __init__() (tensorcv.predicts.predictions.PredictionImage method), 15
 __init__() (tensorcv.predicts.predictions.PredictionScalar method), 15
 __init__() (tensorcv.utils.ssesscreate.NewSessionCreator method), 18
 __init__() (tensorcv.utils.ssesscreate.ReuseSessionCreator method), 18
 _names (tensorcv.callbacks.debug.CheckScalar attribute), 2

A

after_epoch() (tensorcv.callbacks.base.Callback method), 1
 after_inference() (tensorcv.callbacks.inferencer.InferencerBase method), 4
 after_prediction() (tensorcv.predicts.base.Predictor method), 14
 after_reading() (tensorcv.dataflow.base.DataFlow method), 7
 after_run() (tensorcv.callbacks.base.Callback method), 1
 after_run() (tensorcv.callbacks.hooks.Callback2Hook method), 2
 after_run() (tensorcv.callbacks.hooks.Infer2Hook method), 3
 after_run() (tensorcv.callbacks.hooks.Prediction2Hook method), 3
 after_train() (tensorcv.callbacks.base.Callback method), 1
 apply_mask() (in module tensorcv.utils.common), 17
 apply_mask_inverse() (in module tensorcv.utils.common), 17

B

BaseModel (class in tensorcv.models.base), 10
 batch_flatten() (in module tensorcv.models.layers), 11
 batch_norm() (in module tensorcv.models.layers), 11
 before_epoch() (tensorcv.callbacks.base.Callback method), 1
 before_inference() (tensorcv.callbacks.base.Callback method), 1
 before_inference() (tensorcv.callbacks.inferencer.InferencerBase method), 4
 before_read_setup() (tensorcv.dataflow.base.DataFlow method), 7
 before_run() (tensorcv.callbacks.base.Callback method), 1
 before_run() (tensorcv.callbacks.hooks.Callback2Hook method), 2
 before_run() (tensorcv.callbacks.hooks.Infer2Hook method), 3
 before_run() (tensorcv.callbacks.hooks.Prediction2Hook method), 3
 before_train() (tensorcv.callbacks.base.Callback method), 1
 BSDS500 (class in tensorcv.dataflow.dataset.BSDS500), 6
 BSDS500HED (class in tensorcv.dataflow.dataset.BSDS500), 6

C

Callback (class in tensorcv.callbacks.base), 1
 Callback2Hook (class in tensorcv.callbacks.hooks), 2
 Callbacks (class in tensorcv.callbacks.group), 2
 callbacks (tensorcv.predicts.config.PredictConfig attribute), 14
 callbacks (tensorcv.train.config.TrainConfig attribute), 16
 CheckScalar (class in tensorcv.callbacks.debug), 2
 CIFAR (class in tensorcv.dataflow.dataset.CIFAR), 6
 comp_loss_fake() (in module tensorcv.models.losses), 13
 comp_loss_real() (in module tensorcv.models.losses), 13

config (tensorcv.predicts.base.Predictor attribute), 14
 conv() (in module tensorcv.models.layers), 11
 create_graph() (tensorcv.models.base.ModelDes
 method), 10
 create_model() (tensorcv.models.base.ModelDes
 method), 10
 create_session() (tensorcv.utils.ssesscreate.NewSessionCreate
 method), 18
 create_session() (tensorcv.utils.ssesscreate.ReuseSessionCreate
 method), 18

D

d_collection (tensorcv.models.base.GANBaseModel at-
 tribute), 10
 DataFlow (class in tensorcv.dataflow.base), 7
 DataFromFile (class in tensorcv.dataflow.image), 8
 dconv() (in module tensorcv.models.layers), 12
 deconv_size() (in module tensorcv.utils.common), 17
 def_loss() (tensorcv.models.base.GANBaseModel
 method), 10
 default_collection (tensorcv.models.base.BaseModel at-
 tribute), 10
 dense_to_one_hot() (in module ten-
 sorcv.dataflow.common), 7
 dis_callbacks (tensorcv.train.config.GANTrainConfig at-
 tribute), 16
 dropout() (in module tensorcv.models.layers), 12

E

epochs_completed (tensorcv.callbacks.base.Callback at-
 tribute), 1
 epochs_completed (tensorcv.dataflow.base.DataFlow at-
 tribute), 7
 epochs_completed (tensorcv.train.base.Trainer attribute),
 16
 ex_init_model() (tensorcv.models.base.ModelDes
 method), 10

F

fc() (in module tensorcv.models.layers), 12
 FeedInference (class in tensorcv.callbacks.inference), 4
 FeedInferenceBatch (class in ten-
 sorcv.callbacks.inference), 4
 FeedInput (class in tensorcv.callbacks.inputs), 5

G

g_collection (tensorcv.models.base.GANBaseModel at-
 tribute), 11
 GAN_discriminator_loss() (in module ten-
 sorcv.models.losses), 13
 GAN_generator_loss() (in module ten-
 sorcv.models.losses), 13
 GANBaseModel (class in tensorcv.models.base), 10

GANInference (class in tensorcv.callbacks.inference), 4
 GANTrainConfig (class in tensorcv.train.config), 16
 gen_callbacks (tensorcv.train.config.GANTrainConfig at-
 tribute), 16
 get_batch_size() (tensorcv.models.base.ModelDes
 method), 10
 get_data_list() (tensorcv.dataflow.image.ImageDenseLabel
 method), 9
 get_data_list() (tensorcv.dataflow.image.ImageFromFile
 method), 9
 get_data_list() (tensorcv.dataflow.image.ImageLabelFromFolder
 method), 8
 get_default_session_config() (in module ten-
 sorcv.utils.default), 18
 get_discriminator_grads() (ten-
 sorcv.models.base.GANBaseModel method),
 11
 get_discriminator_loss() (ten-
 sorcv.models.base.GANBaseModel method),
 11
 get_discriminator_optimizer() (ten-
 sorcv.models.base.GANBaseModel method),
 11
 get_fetch() (tensorcv.callbacks.inferencer.InferencerBase
 method), 4
 get_file_list() (in module tensorcv.dataflow.common), 7
 get_folder_list() (in module tensorcv.dataflow.common),
 7
 get_folder_names() (in module ten-
 sorcv.dataflow.common), 7
 get_gen_data() (tensorcv.models.base.GANBaseModel
 method), 11
 get_generator_grads() (ten-
 sorcv.models.base.GANBaseModel method),
 11
 get_generator_loss() (ten-
 sorcv.models.base.GANBaseModel method),
 11
 get_generator_optimizer() (ten-
 sorcv.models.base.GANBaseModel method),
 11
 get_global_step (tensorcv.models.base.ModelDes at-
 tribute), 10
 get_global_step (tensorcv.train.base.Trainer attribute), 16
 get_grads() (tensorcv.models.base.BaseModel method),
 10
 get_graph_feed() (tensorcv.models.base.GANBaseModel
 method), 11
 get_graph_feed() (tensorcv.models.base.ModelDes
 method), 10
 get_hooks() (tensorcv.callbacks.group.Callbacks
 method), 2
 get_label_list() (tensorcv.dataflow.image.ImageDenseLabel
 method), 9

get_label_list() (tensorcv.dataflow.image.ImageLabelFromFolder method), 8

get_loss() (tensorcv.models.base.BaseModel method), 10

get_optimizer() (tensorcv.models.base.BaseModel method), 10

get_prediction_placeholder() (tensorcv.models.base.ModelDes method), 10

get_random_vec_placeholder() (tensorcv.models.base.GANBaseModel method), 11

get_rng() (in module tensorcv.utils.utils), 19

get_sample_data() (tensorcv.dataflow.image.DataFromFile method), 8

get_sample_gen_data() (tensorcv.models.base.GANBaseModel method), 11

get_shape2D() (in module tensorcv.models.layers), 12

get_shape4D() (in module tensorcv.models.layers), 13

get_tensors_by_names() (in module tensorcv.utils.common), 17

get_train_placeholder() (tensorcv.models.base.ModelDes method), 10

global_avg_pool() (in module tensorcv.models.layers), 13

global_step (tensorcv.callbacks.base.Callback attribute), 1

H

hooked_sess (tensorcv.predicts.base.Predictor attribute), 14

I

image_overlay() (in module tensorcv.utils.viz), 19

ImageData (class in tensorcv.dataflow.image), 8

ImageDenseLabel (class in tensorcv.dataflow.image), 9

ImageFromFile (class in tensorcv.dataflow.image), 8

ImageLabelFromFile (class in tensorcv.dataflow.image), 8

ImageLabelFromFolder (class in tensorcv.dataflow.image), 8

Infer2Hook (class in tensorcv.callbacks.hooks), 3

InferencerBase (class in tensorcv.callbacks.inferencer), 4

InferImages (class in tensorcv.callbacks.inferencer), 4

InferMat (class in tensorcv.callbacks.inferencer), 4

InferOverlay (class in tensorcv.callbacks.inferencer), 4

InferScalars (class in tensorcv.callbacks.inferencer), 4

input (tensorcv.predicts.base.Predictor attribute), 14

input_val_range() (in module tensorcv.dataflow.common), 7

intensity_to_rgb() (in module tensorcv.utils.viz), 19

L

leaky_relu() (in module tensorcv.models.layers), 13

load_image() (in module tensorcv.dataflow.common), 7

M

main_loop() (tensorcv.train.base.Trainer method), 16

match_tensor_save_name() (in module tensorcv.utils.common), 17

MatlabData (class in tensorcv.dataflow.matlab), 9

max_pool() (in module tensorcv.models.layers), 13

MNIST (class in tensorcv.dataflow.dataset.MNIST), 6

MNISTLabel (class in tensorcv.dataflow.dataset.MNIST), 6

model (tensorcv.predicts.base.Predictor attribute), 14

model_input (tensorcv.models.base.ModelDes attribute), 10

ModelDes (class in tensorcv.models.base), 10

ModelSaver (class in tensorcv.callbacks.saver), 5

Monitors (class in tensorcv.callbacks.monitors), 5

N

new_biases() (in module tensorcv.models.layers), 13

new_normal_variable() (in module tensorcv.models.layers), 13

new_variable() (in module tensorcv.models.layers), 13

new_weights() (in module tensorcv.models.layers), 13

NewSessionCreator (class in tensorcv.utils.sesscreate), 18

next_batch() (tensorcv.dataflow.base.DataFlow method), 7

next_batch() (tensorcv.dataflow.dataset.CIFAR.CIFAR method), 6

next_batch() (tensorcv.dataflow.dataset.MNIST.MNIST method), 6

next_batch() (tensorcv.dataflow.dataset.MNIST.MNISTLabel method), 6

next_batch() (tensorcv.dataflow.image.DataFromFile method), 8

next_batch() (tensorcv.dataflow.image.ImageData method), 8

next_batch() (tensorcv.dataflow.matlab.MatlabData method), 9

next_batch() (tensorcv.dataflow.randoms.RandomVec method), 9

next_batch_dict() (tensorcv.dataflow.base.DataFlow method), 7

next_batch_dict() (tensorcv.dataflow.image.DataFromFile method), 8

P

PeriodicTrigger (class in tensorcv.callbacks.trigger), 5

Prediction2Hook (class in tensorcv.callbacks.hooks), 3

PredictionImage (class in tensorcv.predicts.predictions), 15

PredictionMat (class in tensorcv.predicts.predictions), 15

PredictionMeanScalar (class in tensorcv.predicts.predictions), 15

PredictionOverlay (class in tensorcv.predicts.predictions), 15

PredictionScalar (class in tensorcv.predicts.predictions), 15
 Predictor (class in tensorcv.predicts.base), 14
 PridectConfig (class in tensorcv.predicts.config), 14
 print_warning() (in module tensorcv.dataflow.common), 7
 process_summary() (tensorcv.callbacks.monitors.TFSummaryWriter method), 5
 process_summary() (tensorcv.callbacks.monitors.TrainingMonitor method), 5
 ProxyCallback (class in tensorcv.callbacks.base), 2
 put_fetch() (tensorcv.callbacks.inferencer.InferencerBase method), 4

R

RandomVec (class in tensorcv.dataflow.randoms), 9
 register_callback() (tensorcv.train.base.Trainer method), 16
 register_monitor() (tensorcv.train.base.Trainer method), 16
 reset_epochs_completed() (tensorcv.dataflow.base.DataFlow method), 7
 reset_state() (tensorcv.dataflow.base.DataFlow method), 7
 reset_state() (tensorcv.dataflow.randoms.RandomVec method), 9
 ReuseSessionCreator (class in tensorcv.utils.ssesscreate), 18
 reverse_label_dict() (in module tensorcv.dataflow.common), 7
 RNGDataFlow (class in tensorcv.dataflow.base), 7
 run_predict() (tensorcv.predicts.base.Predictor method), 14

S

save_merge_images() (in module tensorcv.utils.viz), 19
 sess (tensorcv.predicts.base.Predictor attribute), 14
 set_batch_size() (tensorcv.dataflow.base.DataFlow method), 7
 set_batch_size() (tensorcv.models.base.ModelDes method), 10
 set_data_list() (tensorcv.dataflow.image.ImageDenseLabel method), 9
 set_data_list() (tensorcv.dataflow.image.ImageFromFile method), 9
 set_data_list() (tensorcv.dataflow.image.ImageLabelFromFolder method), 8
 set_dropout() (tensorcv.models.base.ModelDes method), 10
 set_is_training() (tensorcv.models.base.ModelDes method), 10
 set_model_input() (tensorcv.models.base.ModelDes method), 10

set_pf() (tensorcv.dataflow.image.ImageFromFile method), 9
 set_prediction_placeholder() (tensorcv.models.base.ModelDes method), 10
 set_train_placeholder() (tensorcv.models.base.ModelDes method), 10
 setup() (tensorcv.dataflow.base.DataFlow method), 7
 setup() (tensorcv.train.base.Trainer method), 16
 setup_graph() (tensorcv.callbacks.base.Callback method), 1
 setup_graph() (tensorcv.train.base.Trainer method), 16
 setup_inferencer() (tensorcv.callbacks.inferencer.InferencerBase method), 4
 setup_summary() (tensorcv.models.base.ModelDes method), 10
 SimpleFeedPredictor (class in tensorcv.predicts.simple), 15
 SimpleFeedTrainer (class in tensorcv.train.simple), 16
 size() (tensorcv.dataflow.base.DataFlow method), 7
 size() (tensorcv.dataflow.dataset.CIFAR.CIFAR method), 6
 size() (tensorcv.dataflow.dataset.MNIST.MNIST method), 6
 size() (tensorcv.dataflow.image.ImageData method), 8
 size() (tensorcv.dataflow.image.ImageFromFile method), 9
 size() (tensorcv.dataflow.image.ImageLabelFromFolder method), 8
 size() (tensorcv.dataflow.matlab.MatlabData method), 9
 size() (tensorcv.dataflow.randoms.RandomVec method), 9
 suffle_data() (tensorcv.dataflow.base.RNGDataFlow method), 7
 suffle_data() (tensorcv.dataflow.image.ImageFromFile method), 9

T

tanh_normalization() (in module tensorcv.dataflow.common), 7
 tensorcv (module), 20
 tensorcv.callbacks (module), 6
 tensorcv.callbacks.base (module), 1
 tensorcv.callbacks.debug (module), 2
 tensorcv.callbacks.group (module), 2
 tensorcv.callbacks.hooks (module), 2
 tensorcv.callbacks.inference (module), 4
 tensorcv.callbacks.inferencer (module), 4
 tensorcv.callbacks.inputs (module), 5
 tensorcv.callbacks.monitors (module), 5
 tensorcv.callbacks.saver (module), 5
 tensorcv.callbacks.summary (module), 5
 tensorcv.callbacks.trigger (module), 5
 tensorcv.dataflow (module), 10
 tensorcv.dataflow.base (module), 7

[tensorcv.dataflow.common \(module\)](#), 7
[tensorcv.dataflow.dataset \(module\)](#), 7
[tensorcv.dataflow.dataset.BSDS500 \(module\)](#), 6
[tensorcv.dataflow.dataset.CIFAR \(module\)](#), 6
[tensorcv.dataflow.dataset.MNIST \(module\)](#), 6
[tensorcv.dataflow.image \(module\)](#), 8
[tensorcv.dataflow.matlab \(module\)](#), 9
[tensorcv.dataflow.randoms \(module\)](#), 9
[tensorcv.models \(module\)](#), 14
[tensorcv.models.base \(module\)](#), 10
[tensorcv.models.layers \(module\)](#), 11
[tensorcv.models.losses \(module\)](#), 13
[tensorcv.predicts \(module\)](#), 16
[tensorcv.predicts.base \(module\)](#), 14
[tensorcv.predicts.config \(module\)](#), 14
[tensorcv.predicts.predictions \(module\)](#), 15
[tensorcv.predicts.simple \(module\)](#), 15
[tensorcv.train \(module\)](#), 17
[tensorcv.train.base \(module\)](#), 16
[tensorcv.train.config \(module\)](#), 16
[tensorcv.train.simple \(module\)](#), 16
[tensorcv.utils \(module\)](#), 20
[tensorcv.utils.common \(module\)](#), 17
[tensorcv.utils.default \(module\)](#), 18
[tensorcv.utils.sesscreate \(module\)](#), 18
[tensorcv.utils.utils \(module\)](#), 19
[tensorcv.utils.viz \(module\)](#), 19
[TFSummaryWriter \(class in tensorcv.callbacks.monitors\)](#),
 5
[train\(\) \(tensorcv.train.base.Trainer method\)](#), 16
[TrainConfig \(class in tensorcv.train.config\)](#), 16
[Trainer \(class in tensorcv.train.base\)](#), 16
[TrainingMonitor \(class in tensorcv.callbacks.monitors\)](#), 5
[TrainSummary \(class in tensorcv.callbacks.summary\)](#), 5
[trigger\(\) \(tensorcv.callbacks.base.Callback method\)](#), 1
[trigger_epoch\(\) \(tensorcv.callbacks.base.Callback](#)
 [method\)](#), 1
[trigger_step\(\) \(tensorcv.callbacks.base.Callback method\)](#),
 2