synthtorch Documentation

Release 0.3.3

Jacob Reinhold

May 08, 2019

Contents:

This package provides models, training, and prediction routines (overwhelmingly) for synthesis tasks, especially in relation to MR and CT brain images. PyTorch is the backend.

CHAPTER 1

Executables

1.1 Neural Network Trainer

train a CNN for MR image synthesis

```
usage: nn-train [-h] -s SOURCE_DIR [SOURCE_DIR ...] -t TARGET_DIR
                [TARGET_DIR ...] [-o TRAINED_MODEL] [-bs BATCH_SIZE] [-c CLIP]
                [-chk CHECKPOINT] [-csv WRITE_CSV] [--disable-cuda]
                [-e {nii,tif,png,jpg}] [-mp] [-l {mse,mae,cp,bce}] [-mg]
                [-n N_JOBS] [-ocf OUT_CONFIG_FILE] [-ps PATCH_SIZE] [-pm]
                [-pl PLOT_LOSS] [-sa SAMPLE_AXIS] [-sd SEED] [-vs VALID_SPLIT]
                [-vsd VALID_SOURCE_DIR [VALID_SOURCE_DIR ...]]
                [-vtd VALID_TARGET_DIR [VALID_TARGET_DIR ...]] [-v]
                [-bt BETAS BETAS] [-nlo]
                [-opt {adam,adamw,sgd,sgdw,nesterov,adagrad,amsgrad,rmsprop}]
                [-wd WEIGHT_DECAY] [-cm {triangular,triangular2,exp_range}]
                [-lrs {cyclic,cosinerestarts}]
                [-mr MOMENTUM_RANGE MOMENTUM_RANGE] [-nc NUM_CYCLES]
                [-rp RESTART_PERIOD] [-tm T_MULT]
                [-ac {relu,lrelu,prelu,elu,celu,selu,tanh,sigmoid}]
                [-dp DROPOUT_PROB] [-eb]
                [-in {normal,xavier,kaiming,orthogonal}] [-ing INIT_GAIN]
                [-ks KERNEL_SIZE] [-lr LEARNING_RATE] [-ne N_EPOCHS]
                [-nl N_LAYERS] [-3d]
                [-na {unet,nconv,vae,segae,densenet,ordnet,lrsdnet,hotnet}]
                [-nm {instance,batch,layer,weight,spectral,none}]
                [-oac {linear,relu,lrelu,prelu,elu,celu,selu,tanh,sigmoid}]
                [-atu] [-at] [-cbp CHANNEL_BASE_POWER] [-ic]
                [-im {nearest,bilinear,trilinear}] [-ns] [-nz NOISE_LVL] [-sp]
                [-sx] [-lrsd LRSD_WEIGHTS LRSD_WEIGHTS] [-ed] [-lp]
                [-ord ORD_PARAMS ORD_PARAMS ORD_PARAMS]
                [-id IMG_DIM [IMG_DIM ...]] [-ls LATENT_SIZE] [-fl]
                [-is INITIALIZE_SEG] [-nseg N_SEG]
                [-li LAST_INIT [LAST_INIT ...]] [-np NORM_PENALTY]
```

(continues on next page)

(continued from previous page)

```
                [-op ORTHO_PENALTY] [-sm SEG_MIN] [-mse] [-mask]
                [-p PROB PROB PROB PROB PROB] [-r ROTATE] [-ts TRANSLATE]
                [-sc SCALE] [-hf] [-vf] [-g GAMMA] [-gn GAIN]
                [-blk BLOCK BLOCK] [-th THRESHOLD] [-pwr NOISE_PWR]
                [-mean MEAN [MEAN ...]] [-std STD [STD ...]] [-tx] [-ty]
```

1.1.1 Required

-s, --source-dir path to directory with source images (multiple paths can be provided for multi-modal synthesis)

-t, --target-dir path to directory with target images (multiple paths can be provided for multi-modal synthesis)

-o, --trained-model path to output the trained model or (if model exists) continue training this model

1.1.2 Options

-bs, --batch-size batch size (num of images to process at once) [Default=5]

Default: 5

-c, --clip gradient clipping threshold [Default=None]

-chk, --checkpoint save the model every *checkpoint* epochs [Default=None]

-csv, --write-csv write the loss to a csv file of this filename [Default=None]

--disable-cuda Disable CUDA regardless of availability

Default: False

-e, --ext Possible choices: nii, tif, png, jpg

extension of training/validation images [Default=None (.nii and .nii.gz)]

-mp, --fp16 enable mixed precision training

Default: False

-l, --loss Possible choices: mse, mae, cp, bce

Use this specified loss function [Default=None, MSE for Unet]

-mg, --multi-gpu use multiple gpus [Default=False]

Default: False

-n, --n-jobs number of CPU processors to use (use 0 if CUDA enabled) [Default=0]

Default: 0

-ocf, --out-config-file output a config file for the options used in this experiment (saves them as a json file with the name as input in this argument)

-ps, --patch-size patch size extracted from image [Default=0, i.e., whole slice or whole image]

Default: 0

-pm, --pin-memory pin memory in dataloader [Default=False]

Default: False

-pl, --plot-loss plot the loss vs epoch and save at the filename provided here [Default=None]

-sa, --sample-axis axis on which to sample for 2d (None for random orientation when NIfTI images given) [Default=2]

Default: 2

-sd, --seed set seed for reproducibility [Default=0]

Default: 0

-vs, --valid-split split the data in source_dir and target_dir into train/validation with this split percentage [Default=0.2]

Default: 0.2

-vsd, --valid-source-dir path to directory with source images for validation, see -vs for default action if this is not provided [Default=None]

-vtd, --valid-target-dir path to directory with target images for validation, see -vs for default action if this is not provided [Default=None]

-v, --verbosity increase output verbosity (e.g., -vv is more than -v)

Default: 0

1.1.3 Optimizer Options

-bt, --betas optimizer parameters (if using SGD, then the first element will be momentum and the second ignored) [Default=(0.9,0.99)]

Default: (0.9, 0.99)

-nlo, --no-load-opt if loading a trained model, do not load the optimizer [Default=False]

Default: False

-opt, --optimizer Possible choices: adam, adamw, sgd, sgdw, nesterov, adagrad, amsgrad, rmsprop

Use this optimizer to train the network [Default=adam]

Default: "adam"

-wd, --weight-decay weight decay parameter for optimizer [Default=0]

Default: 0

1.1.4 Scheduler Options

-cm, --cycle-mode Possible choices: triangular, triangular2, exp_range

type of cycle for cyclic lr scheduler [Default=triangular]

Default: "triangluar"

-lrs, --lr-scheduler Possible choices: cyclic, cosinerestarts

use a learning rate scheduler [Default=None]

-mr, --momentum-range range over which to inversely cycle momentum (does not work w/ all optimizers) [Default=(0.85,0.95)]

Default: (0.85, 0.95)

-nc, --num-cycles number of cycles for cyclic learning rate scheduler [Default=1]

Default: 1

-rp, --restart-period restart period for cosine annealing with restarts [Default=None]

-tm, --t-mult multiplication factor for which the next restart period will extend or shrink (for cosine annealing with restarts) [Default=None]

1.1.5 Neural Network Options

-ac, --activation Possible choices: relu, lrelu, prelu, elu, celu, selu, tanh, sigmoid

type of activation to use throughout network except output [Default=relu]

Default: "relu"

-dp, --dropout-prob dropout probability per conv block [Default=0]

Default: 0

-eb, --enable-bias enable bias calculation in upsampconv layers and final conv layer [Default=False]

Default: False

-in, --init Possible choices: normal, xavier, kaiming, orthogonal

use this type of initialization for the network [Default=kaiming]

Default: "kaiming"

-ing, --init-gain use this initialization gain for initialization [Default=0.2]

Default: 0.2

-ks, --kernel-size convolutional kernel size (squared or cubed) [Default=3]

Default: 3

-lr, --learning-rate learning rate for the optimizer [Default=1e-3]

Default: 0.001

-ne, --n-epochs number of epochs [Default=100]

Default: 100

-nl, --n-layers number of layers to use in network (different meaning per arch) [Default=3]

Default: 3

-3d, --is-3d create a 3d network instead of 2d [Default=False]

Default: False

-na, --nn-arch Possible choices: unet, nconv, vae, segae, densenet, ordnet, lrsdnet, hotnet

specify neural network architecture to use

Default: "unet"

-nm, --normalization Possible choices: instance, batch, layer, weight, spectral, none

type of normalization layer to use in network [Default=instance]

Default: "instance"

-oac, --out-activation Possible choices: linear, relu, lrelu, prelu, elu, celu, selu, tanh, sigmoid

type of activation to use in network on output [Default=linear]

Default: "linear"

1.1.6 UNet Options

-atu, --add-two-up Add two to the kernel size on the upsampling in the U-Net as per Zhao, et al. 2017 [Default=False]

Default: False

-at, --attention use attention gates in up conv layers in unet[Default=False]

Default: False

-cbp, --channel-base-power 2 ** channel_base_power is the number of channels in the first layer and increases in each proceeding layer such that in the n-th layer there are 2 ** (channel_base_power + n) channels [Default=5]

Default: 5

-ic, --input-connect connect the input to the final layers via a concat skip connection [Default=False]

Default: False

-im, --interp-mode Possible choices: nearest, bilinear, trilinear

use this type of interpolation for upsampling [Default=nearest]

Default: "nearest"

-ns, --no-skip do not use skip connections in unet [Default=False]

Default: False

-nz, --noise-lvl add this level of noise to model parameters [Default=0]

Default: 0

-sp, --separable use separable convolutions instead of full convolutions [Default=False]

Default: False

-sx, --softmax use softmax before last layer [Default=False]

Default: False

1.1.7 LRSDNet Options

-lrsd, --lrsd-weights penalties for lrsd [Default=None]

1.1.8 OrdNet/HotNet Options

-ed, --edge use edge map [Default=False]

Default: False

-lp, --laplacian use laplacian [Default=False]

Default: False

-ord, --ord-params ordinal regression params (start, stop, n_bins) [Default=None]

1.1.9 VAE Options

-id, --img-dim if using VAE, then input image dimension must be specified [Default=None]

-ls, --latent-size if using VAE, this controls latent dimension size [Default=2048]

Default: 2048

1.1.10 SegAE Options

-fl, --freeze-last freeze the last layer for training [Default=False]

Default: False

-is, --initialize-seg number of epochs to initialize segmentation layers with seed [Default=0]

Default: 0

-nseg, --n-seg number of segmentation layers [Default=5]

Default: 5

-li, --last-init initial numbers for last layer [Default=None]

-np, --norm-penalty weight for the norm penalty [Default=1]

Default: 1

-op, --ortho-penalty weight for the orthogonality penalty [Default=1]

Default: 1

-sm, --seg-min minimum prob in segmentation [Default=0]

Default: 0

-mse, --use-mse use mse instead of cosine proximity in loss function [Default=False]

Default: False

-mask, --use-mask use brain mask on segmentation and output (during training and testing) [Default=False]

Default: False

1.1.11 Data Augmentation Options

-p, --prob probability of (Affine, Flip, Gamma, Block, Noise) [Default=None]

-r, --rotate max rotation angle [Default=0]

Default: 0

-ts, --translate max fractional translation [Default=None]

-sc, --scale max scale (1-scale,1+scale) [Default=None]

-hf, --hflip horizontal flip [Default=False]

Default: False

-vf, --vflip	vertical flip [Default=False]
	Default: False
-g, --gamma	gamma (1-gamma,1+gamma) for (gain * x ** gamma) [Default=None]
-gn, --gain	gain (1-gain,1+gain) for (gain * x ** gamma) [Default=None]
-blk, --block	insert random blocks of this size range [Default=None]
-th, --threshold	threshold for foreground for blocks, if none use mean [Default=None]
-pwr, --noise-pwr	noise standard deviation/power [Default=0]
	Default: 0
-mean, --mean	normalize input images with this mean (one entry per input directory) [Default=None]
-std, --std	normalize input images with this std (one entry per input directory) [Default=None]
-tx, --tfm-x	apply transforms to x (change this with config file) [Default=True]
	Default: True
-ty, --tfm-y	apply transforms to y [Default=False]
	Default: False

1.2 Neural Network Predictor

The prediction function only supports the use of a configuration file (which can be created from the use of the *nn-train*, see the *–out-config-file* option). This is due to pytorch requiring the parameters to recreate the neural network class, which can then be updated with the trained weights.

Note that you will have to change the *predict_out* and *predict_dir* fields in the .json file with where the output files should be stored and where the source images should come from, respectively.

There may be other fields that need to be altered based on your specific configuration.

CHAPTER 2

Learn

2.1 Layers

synthtorch.learn.layers

define auxillary layers for defining neural networks in pytorch

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Feb 21, 2018

2.2 Learner

synthtorch.learn.learner

train functions for synthtorch neural networks

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Feb 25, 2018

`synthtorch.learn.learner.`**`get_data_augmentation`**(*config: synthtorch.util.config.ExperimentConfig*)

get all data augmentation transforms for training

`synthtorch.learn.learner.`**`get_dataloader`**(*config: synthtorch.util.config.ExperimentConfig*, *tfms: Tuple[List, List] = None*)

get the dataloaders for training/validation

`synthtorch.learn.learner.`**`get_device`**(*disable_cuda=False*)

get the device(s) for tensors to be put on

`synthtorch.learn.learner.`**`get_model`**(*config: synthtorch.util.config.ExperimentConfig*, *enable_dropout: bool = True*, *inplace: bool = False*)

instantiate a model based on an ExperimentConfig class instance

Parameters

- **config** (*ExperimentConfig*) – instance of the ExperimentConfig class
- **enable_dropout** (*bool*) – enable dropout in the model (usually for training)

Returns instance of one of the available models in the synthtorch package

Return type model

2.3 Loss Functions

synthtorch.learn.loss

define loss functions for neural network training

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Feb 20, 2018

2.4 Optimizers

synthtorch.learn.optim

define optimizer auxillary functions for neural network training

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Feb 4, 2018

2.5 Prediction

synthtorch.learn.predict

routines specific to prediction

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Feb 26, 2018

CHAPTER 3

Models

3.1 U-net

synthtorch.models.unet

holds the architecture for a 2d or 3d unet [1,2,3]

References

[1] Ronneberger, Olaf, Philipp Fischer, and Thomas Brox. "U-net: Convolutional networks for biomedical image segmentation." International Conference on Medical image computing and computer-assisted intervention. Springer, Cham, 2015.

[2] O. Cicek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger, "3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation," in Medical Image Computing and Computer-Assisted Intervention (MICCAI), 2016, pp. 424–432.

[3] C. Zhao, A. Carass, J. Lee, Y. He, and J. L. Prince, "Whole Brain Segmentation and Labeling from CT Using Synthetic MR Images," MLMI, vol. 10541, pp. 291–298, 2017.

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Nov 2, 2018

3.2 Variational Autoencoder

synthtorch.models.vae

construct a variational autoencoder

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Jan 29, 2019

3.3 N-layer CNN

synthtorch.models.nconvnet

define the class for a N layer CNN with no max pool, increase in channels, or any of that fancy stuff. This is generally used for testing purposes

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Nov 2, 2018

3.4 SegAE

synthtorch.models.segae

implements a segmentation autoencoder as per [1] with numerous modifications which were empirically found to produce more robust results

References

[1] Atlason, H. E., Love, A., Sigurdsson, S., Gudnason, V., & Ellingsen, L. M. (2018). Unsupervised brain lesion segmentation from MRI using a convolutional autoencoder, 2–4. Retrieved from http://arxiv.org/abs/1811.09655

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Mar 1, 2019

3.5 DenseNet

synthtorch.models.densenet

holds the architecture for a 2d densenet [1] this model is pulled (and modified) from the pytorch repo: https://github.com/pytorch/vision/blob/master/torchvision/models/densenet.py

References

[1] Huang, Gao, et al. "Densely connected convolutional networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Apr 8, 2018

CHAPTER 4

Plot

4.1 Neural Network Visualization Tools

synthtorch.plot.loss

loss visualization plotting tools

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Nov 2, 2018

synthtorch.plot.loss.**plot_loss**(*all_losses: List[list], figsize: Tuple[int, int] = (14, 7), scale: int = 0, ecolor: str = 'red', filename: Optional[str] = None, ax: object = None, label: str = '', plot_error: bool = True*)

plot loss vs epoch for a given list (of lists) of loss values

Parameters

- **all_losses** (*list*) – list of lists of loss values per epoch
- **figsize** (*tuple*) – two ints in a tuple controlling figure size
- **scale** (*int*) – two ints in a tuple controlling figure size
- **ecolor** (*str*) – color of errorbars
- **filename** (*str*) – if provided, save file at this path
- **ax** (*matplotlib ax object*) – supply an ax if desired
- **label** (*str*) – label for ax.plot
- **plot_error** (*bool*) – plot error bars or nah

Returns ax that the plot was created on

Return type ax (matplotlib ax object)

CHAPTER 5

Utilities

5.1 Helper Tools

synthtorch.util.helper

define helper function for defining neural networks in pytorch

Author: Jacob Reinhold (jacob.reinhold@jhu.edu)

Created on: Nov 2, 2018

synthtorch.util.helper.**get_act**(*name: str*, *inplace: bool = True*, *params: Optional[dict] = None*) → Union[<Mock name='mock.nn.modules.activation.ReLU' id='140031288004680'>, <Mock name='mock.nn.modules.activation.LeakyReLU' id='140031288004848'>, <Mock name='mock.nn.modules.activation.Tanh' id='140031288004904'>, <Mock name='mock.nn.modules.activation.Sigmoid' id='140031288004960'>]

get activation module from pytorch must be one of: relu, lrelu, linear, tanh, sigmoid

Parameters

- **name** (*str*) – name of activation function desired
- **inplace** (*bool*) – flag activation to do operations in-place (if option available)
- **params** (*dict*) – dictionary of parameters (as per pytorch documentation)

Returns instance of activation class

Return type act (activation)

synthtorch.util.helper.**get_loss**(*name: str*)

get a loss function by name

synthtorch.util.helper.**get_norm2d**(*name: str*, *num_features: int*, *params: Optional[dict] = None*) → Union[<Mock name='mock.nn.modules.instancenorm.InstanceNorm3d' id='140031288005016'>, <Mock name='mock.nn.modules.batchnorm.BatchNorm3d' id='140031288005800'>]

get a 2d normalization module from pytorch must be one of: instance, batch, none

Parameters

- **name** (*str*) – name of normalization function desired
- **num_features** (*int*) – number of channels in the normalization layer
- **params** (*dict*) – dictionary of optional other parameters for the normalization layer as specified by the pytorch documentation

Returns instance of normalization layer

Return type norm

synthtorch.util.helper.**get_norm3d**(*name: str*, *num_features: int*, *params: Optional[dict] = None*) → Union[<Mock name='mock.nn.modules.instancenorm.InstanceNorm3d' id='140031288005016'>, <Mock name='mock.nn.modules.batchnorm.BatchNorm3d' id='140031288005800'>]

get a 3d normalization module from pytorch must be one of: instance, batch, none

Parameters

- **name** (*str*) – name of normalization function desired
- **num_features** (*int*) – number of channels in the normalization layer
- **params** (*dict*) – dictionary of optional other parameters for the normalization layer as specified by the pytorch documentation

Returns instance of normalization layer

Return type norm

synthtorch.util.helper.**get_optim**(*name: str*)

get an optimizer by name

synthtorch.util.helper.**init_weights**(*net*, *init_type='kaiming'*, *init_gain=0.02*)

Initialize network weights (inspired by https://github.com/junyanz/pytorch-CycleGAN-and-pix2pix/)

Parameters

- **net** (*nn.Module*) – network to be initialized
- **init_type** (*str*) – the name of an initialization method: normal, xavier, kaiming, or orthogonal
- **init_gain** (*float*) – scaling factor for normal, xavier and orthogonal.

Returns None

CHAPTER 6

Indices and tables

- genindex
- modindex

Python Module Index

S

Index

G

I

P

S