A Comparative Analysis of Multi-Task Learning Approaches in the Context of Multi-Label Remote Sensing Image Retrieval

Generated by Doxygen 1.9.2

Chapter 1

A Comparative Analysis of Multi-Task Learning Approaches in the Context of Multi-Label Remote Sensing Image Retrieval

1.1 Table of contents

1.2 General info

This project aims to compare the performance of multi-task approaches in content-based remote sensing image retrieval (CBIR). The goal of all the methods in this work is to learn a metric for multi-label images, such that samples with maximum overlap in label sets are close. The three multi-task methods we compared are:

1. Diverse Visual Feature Aggregation for Deep MetricLearning (Diva) git pdf
2. Divide and Conquer the Embedding Space for MetricLearning (D&C) git pdf
3. Deep Metric Learning with BIER: Boosting Indepen-dent Embedding Robustly (Bier) git pdf

One single-task approach for further comparisons:

1. Graph Relation Network: Modeling Relations Between Scenes for Multilabel Remote-Sensing Image Classification and Retrieval (SNDL) pdf

1.3 Datasets

Remote sensing datasets:

1. BigEarthNet
2. MLRSNet

After downloaded the data, extract them and keep their original structure, then place them in a folder named Ḍataset, for examle, assuming your folder is placed in <$path/Dataset/BigEarthNet>, pass $path/Dataset as input to --source_path

```
Dataset
└──BigEarthNet
|    └──S2A_MSIL2A_20170613T101031_0_48
|           |   S2A_MSIL2A_20170613T101031_0_48_B0
|           |   ...
|    ...
|
└──MLRSNet
|    |   Categories_names.xlsx
|    └──Images
|    |      └──airplane
|    |             |   airplane_00001.jpg
|    |             |   ...
|    |
|    └──labels
|    |      └──airplane.csv
|    ...
```

1.4 Requirements

- python==3.6
- torch==1.7.0
- torchvision==0.8.1
- faiss-gpu==1.6.5
- hypia==0.0.3
- GDAL==3.0.4
- pretrainedmodels==0.7.4
- wandb==0.10.20
- vaex==4.0.0

1.5 Setup

An exemplary setup of a virtual environment containing everything needed:

```
(1) wget  https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
(2) bash Miniconda3-latest-Linux-x86_64.sh (say yes to append path to bashrc)
(3) source .bashrc
(4) conda create -n DL python=3.6
(5) conda activate DL
(6) conda install matplotlib scipy scikit-learn scikit-image tqdm vaex pillow xlrd
(7) conda install pytorch torchvision faiss-gpu cudatoolkit=10.0 -c pytorch
(8) pip install wandb pretrainedmodels hypia
(9) Run the scripts!
```

1.6 Training

Training the baseline is done by using `train_baseline.py` and setting the respective flags, all of which are listed and explained in `parameters.py`.
If you want to train other methods, a set of exemplary runs is provided in `SampleRun.sh`. A basic sample run using default parameters would like this:

```
python train_diva.py --log_online \
                     --dataset MLRSNet  \
                     --source_path ".../Dataset" \
                     --save_path "../Training_Results" \
                     --project MLRSNet --group bier --savename 'bier' \
                     --num_samples_per_class 2  --use_npmem --eval_epoch 10 --nb_epochs 120
```

Here are some notes for training:

- If you want to speed up the data loading from the disk, set the flag `--use_npmem`, it will generate numpy binary file in the path `--source_path`, then the data will be read from these numpy binary file during training.
- During training, metrics listed in `--eval_metric` will be logged for validation/test set. If you also want to log the overlap of embedding distance from intra and inter group, simply set the flag `--is_plot_dist`. A checkpoint is saved for improvements on recall@1 on val set. The default metrics supported are Recall@K, R-Precision@K, MAP@K.
- If a training is stopped accidentally, you can resume the training by set the flag `--load_from_checkpoint`, the training will be restarted from the last checkpoint epoch, and the training results will be written to the original checkpoint folder.

If you want to use W&B to log results during training:

- Create an account here (free): https://wandb.ai
- After the account is set, make sure to include your API key in `parameters.py` under `--wandb_key`.
- Set the flag `--log_online` to use wandb logging, if the network is unavailable in your training environment, set the flag `--wandb_dryrun` to make wandb store the data locally, and you can upload the data with the command `wandb sync <$path/wandb/offline..>`

1.7 Evaluation

Evaluation is done by using `evaluate_model.py` and setting the respective flags, all of which are listed and explained in `evaluate_model.py`. A set of exemplary runs is provided in `SampleRun.sh`. The evaluation results will include a summary of metric scores, png files of retrieved samples, distance density plot of intra and inter group if the flag `--is_plot_dist` is set.

1.8 Implemented Methods

1.8.1 Loss functions

- **Margin loss** [Sampling Matters in Deep Embedding Learning]
- **Binomial loss (boosted)**
- **NCA loss** [Improving Generalization via Scalable Neighborhood Component Analysis]
- **Fast MOCO** Momentum Contrast Loss
- **Adversarial loss**

1.8.2 Batch miner

- **Semihard** [Facenet: A unified embedding for face recognition and clustering]
- **MultiLabelSemihard** [A variation of semihard, take embedding vectors and multi-hot labels as input]
- **Distance** [Sampling Matters in Deep Embeddings Learning]

1.8.3 Architectures

- **ResNet50** [Deep Residual Learning for Image Recognition]

1.8.4 Evaluation Metrics

Metrics based on samples

- **Recall@K**
- **R-Precision@K**
- **MAP@K**

1.9 Contact

Created by Jun Xiang, email: xj.junxiang@gmail.com - feel free to contact me!

Chapter 2

Namespace Index

2.1 Namespace List

Here is a list of all documented namespaces with brief descriptions:

Chapter 3

Hierarchical Index

3.1 Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

Chapter 4

Class Index

4.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 5

Namespace Documentation

5.1 lib.multifeature_resnet50 Namespace Reference

Classes

- class Network

Functions

- def increase_channels (conv, num_channels=None, copy_weights=0)

5.1.1 Detailed Description

```
The network architectures and weights are adapted
and used from  https://github.com/Cadene/pretrained-models.pytorch.
```

5.1.2 Function Documentation

5.1.2.1 increase_channels()

```
def lib.multifeature_resnet50.increase_channels (
              conv,
              num_channels = None,
              copy_weights = 0 )
```

```
 Takes as input a Conv2d layer and returns the Conv2d layer with `num_channels` input channels
    and all the previous weights copied into the new layer.

Args:
    conv (nn.Conv2d):  Conv2d labyer
    num_channels ([type], optional): the new number of input channel. Defaults to None.
    copy_weights (int, optional): copy the weights of the `copy_weights` channel of the old layer
    to the extra channels of the new layer

Returns:
    nn.Conv2d:  Conv2d labyer
```

5.2 utilities.misc Namespace Reference

Classes

- class DataParallel

Functions

- def gimme_params (model)

  ACQUIRE NUMBER OF WEIGHTS ##################.
- def gimme_save_string (config)

  SAVE TRAINING PARAMETERS IN NICE STRING ##################.

5.2.1 Detailed Description

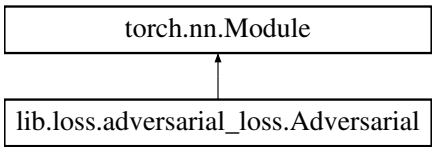
==========

Chapter 6

Class Documentation

6.1 lib.loss.adversarial_loss.Adversarial Class Reference

Inheritance diagram for lib.loss.adversarial_loss.Adversarial:



Public Member Functions

- def __init__ (self, hidden_adversarial_size, direction_dict, decorrnet_lr=0.00001)
- def forward (self, feature_dict)

Public Attributes

- **directions**
- **proj_dim**
- **lr**
- **regressors**

6.1.1 Constructor & Destructor Documentation

6.1.1.1 __init__()

```
def lib.loss.adversarial_loss.Adversarial.__init__ (
              self,
              hidden_adversarial_size,
              direction_dict,
              decorrnet_lr = 0.00001 )

Decorrelation sub embeddings .

Args:
    hidden_adversarial_size (int): the size for hidden layer
    direction_dict (dict): eg.{"discriminative-shared":{'dim':'96-120', 'weight':150}},
    Defaults to 0.00001.
```

6.1.2 Member Function Documentation

6.1.2.1 forward()

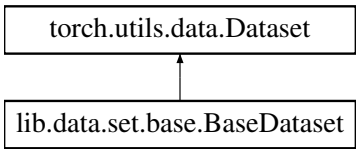
```
def lib.loss.adversarial_loss.Adversarial.forward (
              self,
              feature_dict )

Args:
    feature_dict (dict): eg.{'discriminative': sub_embeddings1, 'shared': sub_embeddings2},
                         sub_embeddings1 [batch_size, dim1], sub_embeddings2 [batch_size, dim2]
```

The documentation for this class was generated from the following file:

- code/lib/loss/adversarial_loss.py

6.2 lib.data.set.base.BaseDataset Class Reference

Inheritance diagram for lib.data.set.base.BaseDataset:



Public Member Functions

- def __init__ (self, image_list, dataset_name, npmem_file="", conversion=None, transform=None, is_↩ training=False, dset_type='train', include_aux_augmentations=False)
- def **__len__** (self)
- def **nb_classes** (self)
- def **__getitem__** (self, index)
- def **get_label** (self, index)
- def **set_subset** (self, subset_indices)
- def process_image (self, img)

Public Attributes

- **dataset_name**
- **transform**
- **dset_type**
- **is_training**
- **npmem_file**
- **include_aux_augmentations**
- **conversion**
- **ys**
- **image_dict**
- **l**

6.2.1 Constructor & Destructor Documentation

6.2.1.1 __init__()

```
def lib.data.set.base.BaseDataset.__init__ (
              self,
              image_list,
              dataset_name,
              npmem_file = "",
              conversion = None,
              transform = None,
              is_training = False,
              dset_type = 'train',
              include_aux_augmentations = False )
```

```
create dataset .

Args:
    image_list (list): contains file_paths  and multi-hot labels
    dataset_name (str): choose from {"MLRSNet", "BigEarthNet}
    npmem_file (str, optional): the path of npmem_file, if set use_npmem true,
                                it will be automatically generated
    conversion (dict, optional): dictionary, {'label': label_name}
    transform (dict, optional): keys: sz_crop, input_shape. Defaults to None.
    is_training (bool, optional):if set, apply random flip and crop to the data split,
                                 else apply center crop. Defaults to False.
    dset_type (str, optional): select from {'train','val,'test'}. Defaults to 'train'.
    include_aux_augmentations (bool, optional): if set true, apply rotation to
                                get augmented image data. Defaults to False.
```

6.2.2 Member Function Documentation

6.2.2.1 process_image()

```
def lib.data.set.base.BaseDataset.process_image (
              self,
              img )
```

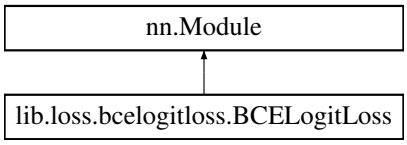
```
Preprocessing images .
For training this function randomly crop images and
flips the image randomly.
For testing we use the center crop of the image.

Args:
    img (numpy array)

Returns:
    torch.Tensor: shape [12, 100, 100] for BigEarthNet, [3, 224, 224] for MLRSNet
```

The documentation for this class was generated from the following file:

- code/lib/data/set/base.py

6.3 lib.loss.bcelogitloss.BCELogitLoss Class Reference

Inheritance diagram for lib.loss.bcelogitloss.BCELogitLoss:



Public Member Functions

- def __init__ (self, embed_dim, num_labels, bce_lr=0.00001, weight=0)
- def forward (self, feature, label)

Public Attributes

- **num_labels**
- **regressor**
- **lr**

6.3.1 Constructor & Destructor Documentation

6.3.1.1 __init__()

```
def lib.loss.bcelogitloss.BCELogitLoss.__init__ (
              self,
              embed_dim,
              num_labels,
              bce_lr = 0.00001,
              weight = 0 )


Args:
    embed_dim:  the size of embedding
    num_labels: the number of classes
```

6.3.2 Member Function Documentation

6.3.2.1 forward()

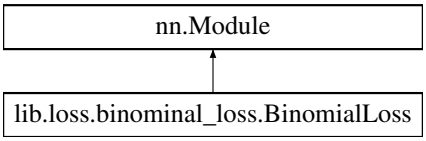
```
def lib.loss.bcelogitloss.BCELogitLoss.forward (
              self,
              feature,
              label )


Args:
    feature: tensor, shape [batchsize x embed_dim]
    label: tensor, shape [batchsize x multihot], like [1,0,1,0,1,1,1]
```

The documentation for this class was generated from the following file:

- code/lib/loss/bcelogitloss.py

6.4 lib.loss.binominal_loss.BinomialLoss Class Reference

Inheritance diagram for lib.loss.binominal_loss.BinomialLoss:



Public Member Functions

- def __init__ (self, C=25, alpha=2.0, beta=0.5, eta_style=True, beta_lr=0.0005, is_beta_trainable=False, ∗∗kwargs)
- def forward (self, normed_fvecs, T)

Public Attributes

- **C**
- **alpha**
- **eta_style**
- **initial_acts**
- **shrinkage**
- **is_beta_trainable**
- **beta**
- **beta_lr**
- **nu**

6.4.1 Constructor & Destructor Documentation

6.4.1.1 __init__()

```
def lib.loss.binominal_loss.BinomialLoss.__init__ (
              self,
              C = 25,
              alpha = 2.0,
              beta = 0.5,
              eta_style = True,
              beta_lr = 0.0005,
              is_beta_trainable = False,
             ** kwargs )

Boost a bionminal loss .
Implement according to paper: https://arxiv.org/abs/1801.04815

Args:
    C (int, optional): cost for neagive pairs. Defaults to 25.
    alpha (float, optional): the scaling parameter . Defaults to 2.0.
    beta (float, optional): margin for binomial, Defaults to 0.5.
    eta_style (bool, optional): [description]. Defaults to True.
    beta_lr (float, optional): learning rate. Defaults to 0.0005.
    is_beta_trainable (bool, optional): . Defaults to False.
```

6.4.2 Member Function Documentation

6.4.2.1 forward()

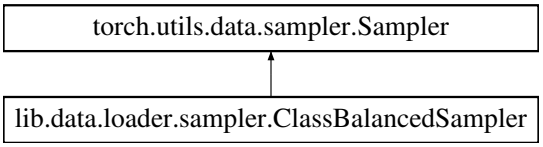
```
def lib.loss.binominal_loss.BinomialLoss.forward (
              self,
              normed_fvecs,
              T )

Compute the forward pass of the model .

Args:
    normed_fvecs (dict):multi-feature dictionary,
                        each value contains sub embeddings with shape [batchsize x sub embedding size]
    T (tensor): category labels, shape(batchsize, )
```

The documentation for this class was generated from the following file:

- code/lib/loss/binominal_loss.py

6.5 lib.data.loader.sampler.ClassBalancedSampler Class Reference

Inheritance diagram for lib.data.loader.sampler.ClassBalancedSampler:



Public Member Functions

- def __init__ (self, num_images, image_dict, num_samples_per_class=2)
- def **__iter__** (self)
- def **__len__** (self)

Public Attributes

- **image_dict**
- **samples_per_class**
- **sampler_length**

6.5.1 Detailed Description

```
Sampler that generates class balanced indices .
For example, choosing batch_size = 50 and nun_samples_per_class = 2
will result in
50 indices, which point to 2 samples from 50/2=25 randomly picked classes.

Yields:
    list: indexes of images
```

6.5.2 Constructor & Destructor Documentation

6.5.2.1 __init__()

```
def lib.data.loader.sampler.ClassBalancedSampler.__init__ (
              self,
              num_images,
              image_dict,
              num_samples_per_class = 2 )

Initialize the class .

Args:
    num_images (int):
    image_dict (dict): key: class labels, values: the index of images
    num_samples_per_class (int, optional):  Defaults to 2.
```

The documentation for this class was generated from the following file:

- code/lib/data/loader/sampler.py

6.6 utilities.logger.CSV_Writer Class Reference

Public Member Functions

- def **__init__** (self, save_path)
- def **log** (self, group, segments, content)

Public Attributes

- **save_path**
- **written**
- **n_written_lines**

6.6.1 Detailed Description

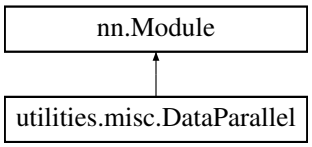
```
Writes CSV writer .
```

The documentation for this class was generated from the following file:

- code/utilities/logger.py

6.7 utilities.misc.DataParallel Class Reference

Inheritance diagram for utilities.misc.DataParallel:



Public Member Functions

- def **__init__** (self, model, device_ids, dim)
- def **forward** (self, x)

Public Attributes

- **model**
- **network**

The documentation for this class was generated from the following file:

- code/utilities/misc.py

6.8 lib.loss.batchminner.distance.Distance Class Reference

Public Member Functions

- def **__init__** (self, lower_cutoff=0.5, upper_cutoff=1.4)
- def __call__ (self, batch, labels)
- def **inverse_sphere_distances** (self, batch, anchor_to_all_dists, labels, anchor_label)
- def **pdist** (self, A)

Public Attributes

- **lower_cutoff**
- **upper_cutoff**

6.8.1 Member Function Documentation

6.8.1.1 __call__()

```
def lib.loss.batchminner.distance.Distance.__call__ (
              self,
              batch,
              labels )
```

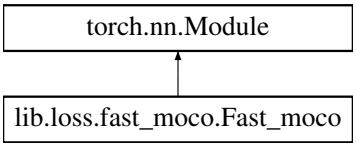
```
 Generate tripets based on category labels

Args:
    batch (tensor):  dim [batchsize,embedding dim]
    labels (tensor): category labels, dim [batchsize,1]

Returns:
    tuple: index [anchor, positive, negative]
```

The documentation for this class was generated from the following file:

- code/lib/loss/batchminner/distance.py

6.9 lib.loss.fast_moco.Fast_moco Class Reference

Inheritance diagram for lib.loss.fast_moco.Fast_moco:



Public Member Functions

- def **__init__** (self, config, moco_temperature=0.07, moco_momentum=0.9, moco_n_key_batches=30, lower_cutoff=0.5, upper_cutoff=1.4)
- def **update_memory_queue** (self, embeddings)
- def create_memory_queue (self, model, dataloader, device, opt_key=None)
- def **shuffleBN** (self, bs)
- def forward (self, query_batch, key_batch)

Public Attributes

- **temperature**
- **momentum**
- **n_key_batches**
- **lower_cutoff**
- **upper_cutoff**
- **diva_features**
- **sz_embedding**
- **memory_queue**
- **n_keys**
- **reference_labels**

6.9.1 Member Function Documentation

6.9.1.1 create_memory_queue()

```
def lib.loss.fast_moco.Fast_moco.create_memory_queue (
              self,
              model,
              dataloader,
              device,
              opt_key = None )

Create a memory queue for the given model
```

6.9.1.2 forward()

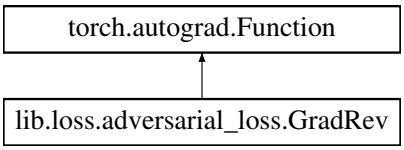
```
def lib.loss.fast_moco.Fast_moco.forward (
              self,
              query_batch,
              key_batch )

Args:
    query_batch: embeddings from images, torch.Tensor (BS x DIM)
    key_batch: embeddings from augumented images, torch.Tensor(BS x DIM)
```

The documentation for this class was generated from the following file:

- code/lib/loss/fast_moco.py

6.10 lib.loss.adversarial_loss.GradRev Class Reference

Inheritance diagram for lib.loss.adversarial_loss.GradRev:



Static Public Member Functions

- def forward (self, x)
- def backward (self, grad_output)

6.10.1 Detailed Description

```
Implements an autograd class to flip gradients during backward pass.
```

6.10.2 Member Function Documentation

6.10.2.1 backward()

```
def lib.loss.adversarial_loss.GradRev.backward (
              self,
              grad_output )  [static]
```

```
Container to reverse gradient signal during backward pass.

Input:
    grad_output: any computed gradient.
```

6.10.2.2 forward()

```
def lib.loss.adversarial_loss.GradRev.forward (
              self,
              x )  [static]
```

```
Container which applies a simple identity function.

Input:
    x: any torch tensor input.
```

The documentation for this class was generated from the following file:

- code/lib/loss/adversarial_loss.py

6.11 utilities.logger.InfoPlotter Class Reference

Public Member Functions

- def **__init__** (self, save_path, title='Training Log', figsize=(25, 19))
- def **make_plot** (self, base_title, title_append, sub_plots, sub_plots_data)

Public Attributes

- **save_path**
- **title**
- **figsize**
- **colors**
- **ov_title**

6.11.1 Detailed Description

```
PLOT SUMMARY IMAGE
```

The documentation for this class was generated from the following file:

- code/utilities/logger.py

6.12 lib.loss.batchminner.intra_random.Intra_random Class Reference

Public Member Functions

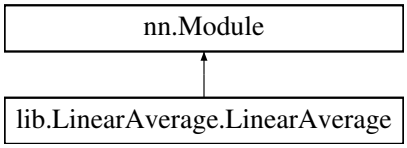
- def **__call__** (self, batch, labels)

The documentation for this class was generated from the following file:

- code/lib/loss/batchminner/intra_random.py

6.13 lib.LinearAverage.LinearAverage Class Reference

Inheritance diagram for lib.LinearAverage.LinearAverage:



Public Member Functions

- def __init__ (self, embed_dim, N, T=0.05, momentum=0.5)
- def forward (self, embed, indexes)

Public Attributes

- **nLem**

6.13.1 Constructor & Destructor Documentation

6.13.1.1 __init__()

```
def lib.LinearAverage.LinearAverage.__init__ (
              self,
              embed_dim,
              N,
              T = 0.05,
              momentum = 0.5 )
```

```
Build the memory bank for Scalable Neighborhood Component Analysis

Args:
    embed_dim (int): dimension of embedding vector
    N (int): the length of dataset
    T (float, optional): temperature, Defaults to 0.05.
    momentum (float, optional): momentum for non-parametric updates. Defaults to 0.5.
```

6.13.2 Member Function Documentation

6.13.2.1 forward()

```
def lib.LinearAverage.LinearAverage.forward (
              self,
              embed,
              indexes )
```

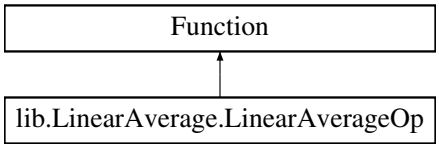
```
Returns the similarity matrices of the given embeddings .

Args:
    embed (torch tensor): embeddings of the mini batch
    indexes (list): index of the mini batch

Returns:
    torch tensor: the similarity mat between the mini-batch embeddings and the memory bank
```

The documentation for this class was generated from the following file:

- code/lib/LinearAverage.py

6.14 lib.LinearAverage.LinearAverageOp Class Reference

Inheritance diagram for lib.LinearAverage.LinearAverageOp:



Static Public Member Functions

- def forward (self, embed, indexes, memory, params)
- def **backward** (self, gradOutput)

6.14.1 Member Function Documentation

6.14.1.1 forward()

```
def lib.LinearAverage.LinearAverageOp.forward (
              self,
              embed,
              indexes,
              memory,
              params )  [static]
```

```
Perform forward pass of the mini batch

Args:
    embed (torch tensor): embeddings of the mini batch
    indexes (list):  shape (N, embeding dim)
    memory (torch tensor): memory bank
    params (float): temperature and momentum

Returns:
    torch tensor: the similarity mat between the mini-batch embeddings and the memory bank
```

The documentation for this class was generated from the following file:

- code/lib/LinearAverage.py

6.15 utilities.logger.LOGGER Class Reference

Public Member Functions

- def __init__ (self, config, sub_loggers=[ ], prefix=None, start_new=True, log_online=False)
- def **update** (self, ∗sub_loggers, all=False)

Public Attributes

- **config**
- **prefix**
- **sub_loggers**
- progress_saver

  Make Logging Directories.
- save_path

  WandB Init.
- **log_online**

6.15.1 Constructor & Destructor Documentation

6.15.1.1 __init__()

```
def utilities.logger.LOGGER.__init__ (
              self,
              config,
              sub_loggers = [],
              prefix = None,
              start_new = True,
              log_online = False )
```

```
LOGGER Internal Structure:

self.progress_saver: Contains multiple Progress_Saver instances to log metrics
                     for main metric subsets (e.g. "Train" for training metrics)
                     ['main_subset_name']: Name of each main subset (-> e.g. "Train")
                     .groups: Dictionary of subsets belonging to one of the main subsets,
                     e.g. ["Recall", "NMI", ...]
                    ['specific_metric_name']: Specific name of the metric of interest, e.g. Recall@1.
```

6.15.2 Member Data Documentation

6.15.2.1 progress_saver

```
utilities.logger.LOGGER.progress_saver
```

Make Logging Directories.

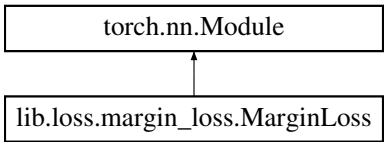
Set Graph and CSV writer

The documentation for this class was generated from the following file:

- code/utilities/logger.py

6.16 lib.loss.margin_loss.MarginLoss Class Reference

Inheritance diagram for lib.loss.margin_loss.MarginLoss:



Public Member Functions

- def __init__ (self, nb_classes, beta=1.2, beta_lr=0.0005, margin=0.2, nu=0.1, is_beta_trainable=True, class↩ _specific_beta=False, batchminner=None, ∗∗kwargs)
- def forward (self, feature, labels)

Public Attributes

- **nb_classes**
- **class_specific_beta**
- **is_beta_trainable**
- **beta**
- **beta_lr**
- **nu**
- **margin**
- **batchminner**

6.16.1 Constructor & Destructor Documentation

6.16.1.1 __init__()

```
def lib.loss.margin_loss.MarginLoss.__init__ (
              self,
              nb_classes,
              beta = 1.2,
              beta_lr = 0.0005,
              margin = 0.2,
              nu = 0.1,
              is_beta_trainable = True,
              class_specific_beta = False,
              batchminner = None,
             ** kwargs )

Initialize the class .

Args:
    nb_classes (int):  Number of classes in the train dataset.
                       Used to initialize class-specific boundaries beta
    beta (float, optional): margin beta. Defaults to 1.2.
    beta_lr (float, optional): learning rate for beta. Defaults to 0.0005.
    margin (float, optional): Margin between positive and negative pairs. Defaults to 0.2.
    nu (float, optional): Regularisation Parameter for beta values if they are learned. Defaults to 0.1.
    is_beta_trainable (bool, optional): if set, beta is trainable. Defaults to True.
    class_specific_beta (bool, optional): if set, beta is trainable for each class. Defaults to False.
    batchminner (optional):  Defaults to None.
```

6.16.2 Member Function Documentation

6.16.2.1 forward()

```
def lib.loss.margin_loss.MarginLoss.forward (
              self,
              feature,
              labels )
```

```
Calculate the loss .

Args:
    feature (tensor): embedding vectors, shape(batchsize x sub embedding size)
    labels (tensor):  if batchminner is multiLabelSemihard, shape(batchsize, L);
                      else category labels shape(batchsize, L)
```

The documentation for this class was generated from the following file:

- code/lib/loss/margin_loss.py

6.17 lib.faissext.MemoryReserver Class Reference

Public Member Functions

- def **__init__** (self)
- def **lock** (self, backend)
- def **release** (self)

Public Attributes

- **memory_holder**

6.17.1 Detailed Description

```
Vaniss memory manager .
```

The documentation for this class was generated from the following file:

- code/lib/faissext.py

6.18 lib.loss.batchminner.multiLabel_semihard.MultiLabelSemihard Class Reference

Public Member Functions

- def **__init__** (self, max_negatives_per_pos=3, max_trips_per_anchor=3)
- def __call__ (self, batch, labels)
- def **pdist** (self, A)

Public Attributes

- **max_negatives_per_pos**
- **max_trips_per_anchor**

6.18.1 Detailed Description

```
MultiLabel semihard.
   reference: https://github.com/Junx0924/multilabel-deep-metric/blob/master/src/utils.py.
```

6.18.2 Member Function Documentation

6.18.2.1 __call__()

```
def lib.loss.batchminner.multiLabel_semihard.MultiLabelSemihard.__call__ (
              self,
              batch,
              labels )
```

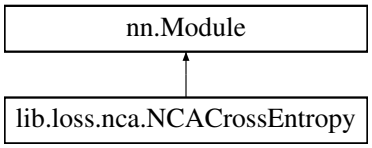
```
 Generate tripets based on multi-hot labels
Args:
    batch (tensor):  dim [batchsize,embedding dim]
    labels (tensor): multi-hot labels, dim [batchsize,L]
Returns:
    tuple: index [anchor, positive, negative]
```

The documentation for this class was generated from the following file:

- code/lib/loss/batchminner/multiLabel_semihard.py

6.19 lib.loss.nca.NCACrossEntropy Class Reference

Inheritance diagram for lib.loss.nca.NCACrossEntropy:



Public Member Functions

- def __init__ (self, labels, margin=0)
- def forward (self, embed_sim, indexes)

Public Attributes

- **labels_sim**
- **margin**

6.19.1 Detailed Description

```
    Store all the labels of the dataset.
    Only pass the indexes of the training instances during forward.
```

6.19.2 Constructor & Destructor Documentation

6.19.2.1 __init__()

```
def lib.loss.nca.NCACrossEntropy.__init__ (
              self,
              labels,
              margin = 0 )

Initializes labels for a training dataset tensor

Args:
    labels (tensor): all the labels for training dataset, shape [len(training dataset),num_classes],
                     multi-hot encoding like [1,0,1,0]
    margin (int, optional): classification margin. Defaults to 0.
```

6.19.3 Member Function Documentation

6.19.3.1 forward()

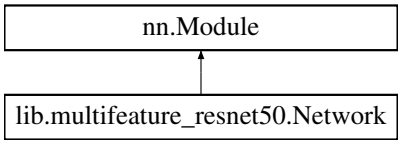
```
def lib.loss.nca.NCACrossEntropy.forward (
              self,
              embed_sim,
              indexes )

Args:
    embed_sim ( tensor): shape [batchsize x len(training dataset)],
                         the similarity mat between the mini-batch embeddings and the memory bank
    indexes (list): indexes for the mini-batch
```

The documentation for this class was generated from the following file:

- code/lib/loss/nca.py

6.20 lib.multifeature_resnet50.Network Class Reference

Inheritance diagram for lib.multifeature_resnet50.Network:



Public Member Functions

- def **__init__** (self, config)
- def forward (self, x, use_penultimate=False)

Public Attributes

- **feature_dim**
- **features**
- **features_pooling**
- **features_dropout**
- **last_linear**
- **parameters_dict**

6.20.1 Member Function Documentation

6.20.1.1 forward()

```
def lib.multifeature_resnet50.Network.forward (
              self,
              x,
              use_penultimate = False )


Performs a forward pass through the network .

Args:
    x (torch tensor):
    use_penultimate (bool, optional):

Returns:
    torch tensor: if use_penultimate set true,
                  return normalized vectors of last linear layer, Defaults to False.
    or dict: if use_penultimate set false,
             return dict of normalized vectors of embedding layer
```

The documentation for this class was generated from the following file:

- code/lib/multifeature_resnet50.py

6.21 utilities.logger.Progress_Saver Class Reference

Public Member Functions

- def **__init__** (self)
- def **log** (self, segment, content, group=None)

Public Attributes

- **groups**

The documentation for this class was generated from the following file:

- code/utilities/logger.py

6.22 lib.loss.batchminner.random_distance.Random_distance Class Reference

Public Member Functions

- def **__init__** (self, lower_cutoff=0.5, upper_cutoff=1.4)
- def **__call__** (self, batch, labels)
- def **inverse_sphere_distances** (self, batch, anchor_to_all_dists, labels, anchor_label)
- def **pdist** (self, A)

Public Attributes

- **lower_cutoff**
- **upper_cutoff**

The documentation for this class was generated from the following file:

- code/lib/loss/batchminner/random_distance.py

6.23 lib.loss.batchminner.semihard.Semihard Class Reference

Public Member Functions

- def **__call__** (self, batch, labels)
- def **pdist** (self, A)

The documentation for this class was generated from the following file:

- code/lib/loss/batchminner/semihard.py

Index