
PaddleHelix

Release 1.0.0

2021, Baidu Inc.

Dec 30, 2022

OVERVIEW

1	Installation	3
1.1	OS support	3
1.2	Python version	3
1.3	Dependencies	3
1.4	Quick Start	3
2	Tutorials	5
3	Examples	7
4	Guide for developers	9
5	Contribution	11
5.1	Installation guide	11
5.2	Tutorials	13
5.3	Guide for developers	15
5.4	Contact Us	16
5.5	pahelix.datasets	16
5.6	pahelix.featurizers	31
5.7	pahelix.model_zoo	33
5.8	pahelix.networks	36
5.9	pahelix.utils	41
	Python Module Index	49
	Index	51



INSTALLATION

1.1 OS support

Windows, Linux and OSX

1.2 Python version

Python 3.6, 3.7

1.3 Dependencies

- PaddlePaddle >= 2.0.0rc0
- pgl >= 2.1

1.4 Quick Start

- PaddleHelix can be installed directly with pip:

```
$ pip install paddlehelix
```

- or install from source:

```
$ pip install --upgrade git+https://github.com/PaddlePaddle/PaddleHelix.git
```

Note: Please check our [Installation guide](#) part for full installation prerequisites and guide.

TUTORIALS

- We provide abundant *Tutorials* to help you navigate the repository and start quickly.
- PaddleHelix is based on [PaddlePaddle](#), a high-performance Parallelized Deep Learning Platform.

EXAMPLES

- Compound Representation Learning and Property Prediction
- Protein Representation Learning and Property Prediction
- Drug Target Interaction
- Molecular Generation
- Drug Drug Synergy
- LinearRNA

GUIDE FOR DEVELOPERS

- If you need help in modifying the source code of **PaddleHelix**, please see our [Guide for developers](#).

CONTRIBUTION

If you would like to develop and maintain PaddleHelix with us, please refer to our [GitHub repo](#).

5.1 Installation guide

Table of Contents

- *Installation guide*
 - *Prerequisites*
 - *Dependencies*
 - *Instruction*

5.1.1 Prerequisites

- OS support: Windows, Linux and OSX
- Python version: **3.6, 3.7**

5.1.2 Dependencies

(- means no specific version requirement for that package)

Name	Version
numpy	-
pandas	-
networkx	-
paddlepaddle	>=2.0.0rc0
pgl	>=2.1
rdkit	-
sklearn	-

5.1.3 Instruction

Since PaddleHelix depends on the `paddlepaddle` of version 2.0.0rc0 or above, and `rdkit` cannot be installed directly using `pip`, we suggest using `conda` to create a new environment for the installation. Detailed instruction is shown below:

- If you do not have `conda` installed, please [install](#) it at first:
- Create a new environment with `conda`:

```
$ conda create -n paddlehelix python=3.7
```

- Activate the environment just created:

```
$ conda activate paddlehelix
```

- Install `rdkit` using `conda`:

```
$ conda install -c conda-forge rdkit
```

- Install the right version of `paddlepaddle` according to the device (CPU/GPU) you want to run PaddleHelix on.

1) If you want to use the GPU version of `paddlepaddle`, run this:

```
$ python -m pip install paddlepaddle-gpu -f https://paddlepaddle.org.cn/whl/stable.html
```

2) Or if you want to use the CPU version of `paddlepaddle`, run this:

```
$ python -m pip install paddlepaddle -i https://mirror.baidu.com/pypi/simple
```

Note: The version of `paddlepaddle` should be higher than **2.0**. Check [paddlepaddle official document](#) for more installation guide.

- Install `ppl` using `pip`:

```
$ pip install ppl
```

- Install **PaddleHelix** using `pip`:

```
$ pip install paddlehelix
```

- The installation is done!

Note: After playing, if you want to deactivate the `conda` environment, do this:

```
$ conda deactivate
```


5.2 Tutorials

Table of Contents

- *Tutorials*
 - *Backgrounds*
 - *Navigating PaddleHelix*
 - *Tutorials*
 - *Run tutorials locally*

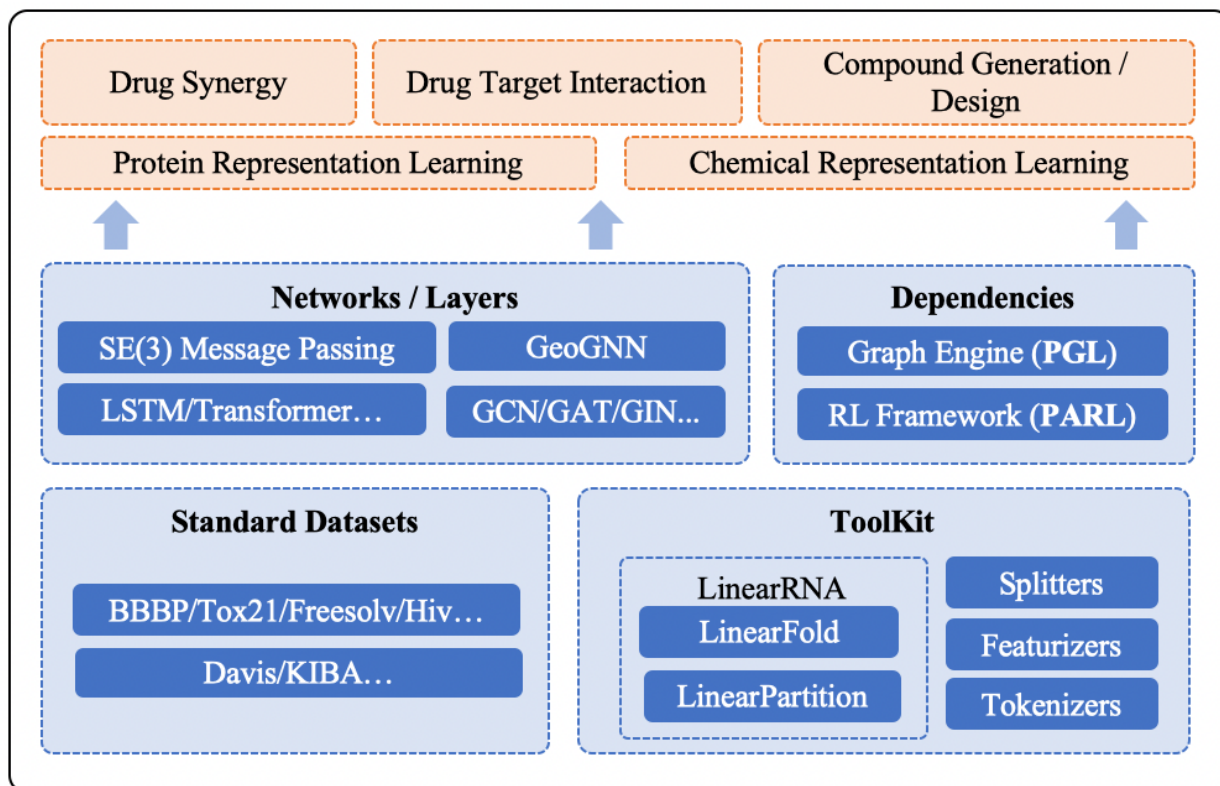
5.2.1 Backgrounds

Machine learning (ML), especially deep learning (DL), is playing an increasingly important role in the pharmaceutical industry and bio-informatics. For instance, the DL-based methodology is found to predict the [drug target interaction](#) and [molecule properties](#) with reasonable precision and quite low computational cost, while those properties can only be accessed through *in vivo*/*in vitro* experiments or computationally expensive simulations (molecular dynamics simulation etc.) before. As another example, *in silico* [RNA folding](#) and [protein folding](#) are becoming more likely to be accomplished with the help of deep neural models. The usage of ML and DL can greatly improve efficiency, and thus reduce the cost of drug discovery, vaccine design, etc.

In contrast to the powerful ability of DL metrics, a key challenge lying in utilizing them in the drug industry is the contradiction between the demand for huge data for training and the limited annotated data. Recently, there is a tremendous success in adopting self-supervised learning in natural language processing and computer vision, showing that a large corpus of unlabeled data can be beneficial to learning universal tasks. In molecule representations, there is a similar situation. We have a large amount of unlabeled data, including protein sequences (over 100 million) and compounds (over 50 million) but relatively small annotated data. It is quite promising to adopt the DL-based pre-training technique in the representation learning of chemical compounds, proteins, RNA, etc.

PaddleHelix is a high-performance ML-based bio-computing framework. It features large-scale representation learning and easy-to-use APIs, providing pharmaceutical and biological researchers and engineers convenient access to the most up-to-date and state-of-the-art AI tools.

5.2.2 Navigating PaddleHelix



5.2.3 Tutorials

- Predicting Drug Target Interaction_GraphDTA, MolTrans
- Compound Representation Learning and Property Prediction
- Protein Representation Learning and Property Prediction
- Molecular Generation
- Predicting RNA Secondary Structure

5.2.4 Run tutorials locally

The tutorials are written as **Jupyter Notebooks** and designed to be smoothly run on your own machine. If you don't have Jupyter installed, please refer to [here](#). And please also install **PaddleHelix** before proceeding (*Installation guide*).

After the installation of **Jupyter**, please go through the following steps:

1. Clone this repository to your own machine
2. Change the working directory of your shell to `path_to_your_repo/PaddleHelix/tutorials/`
3. Open Jupyter lab with the command `jupyter-lab`, wait for your web browser being called out
4. All the tutorials should be in the File Browser now, click and enjoy!

5.3 Guide for developers

If you need to modify the algorithms/models in **PaddleHelix**, you have to switch to the developer mode. The core algorithms of **PaddleHelix** are mostly implemented in Python, but some also in C++, so you cannot develop **PaddleHelix** simply with `pip install --editable {pahelix_path}`. To develop on your machine, please do the following:

- Please follow the [Installation guide](#) part to install all dependencies of **PaddleHelix** (paddlepaddle >= 2.0.0rc0, pgl >= 2.1).
- If you have already installed distributed **PaddleHelix** with `pip install paddlehelix`, please uninstall it with:

```
$ pip uninstall paddlehelix
```

- Clone this repository to your local machine, supposed path at `/path_to_your_repo/`:

```
$ git clone https://github.com/PaddlePaddle/PaddleHelix.git /path_to_your_repo/
```

```
$ cd /path_to_your_repo/
```

- Depends on which model you'd like to modify, go to **LinearRNA** or **Other algorithms**:

LinearRNA

The source code of LinearRNA is at `./c/pahelix/toolkit/linear_rna/linear_rna`. You could modify it for your needs. Then remember to return to the root directory of the repository, run scripts below to re-compile (please ensure there are `cmake >= 3.6` and `g++ >= 4.8` on your machine):

```
$ sh scripts/prepare.sh
```

```
$ sh scripts/build.sh
```

- After a successful compilation, *import* LinearRNA as following:

```
$ cd build
```

```
$ python
```

```
>>> import c.pahelix.toolkit.linear_rna.linear_rna as linear_rna
```

- Except LinearRNA, other algorithms in PaddleHelix are all implemented in Python.

Other algorithms

If you want to change these algorithms, just find and modify corresponding `.py` files under the path `./pahelix`, then add `/path_to_your_repo/` to your Python environment path:

```
import sys
sys.path.append('/path_to_your_repo/')
import pahelix
```

- If you have any question or suggestion, feel free to file on our [GitHub issue page](#). We will response **ASAP**.

5.4 Contact Us

5.4.1 Bug Reports

You can file bug reports on our [GitHub issue page](#), and they will be addressed **ASAP**.

Note: Reporting Issues

When reporting a bug, please include detailed information that will help us solve the issue. See sample format as below:

- Issue name
 - URL
 - Your contact information
 - Expected result
 - Actual result
 - Action taken
-

5.4.2 Join Us

If you have any questions and concern, or if you want to contribute together with us, please join QQ group: 699105483

We are available 24/7 and we will get back to you **ASAP**!

5.5 pahelix.datasets

Table of Contents

- *pahelix.datasets*
 - *bace_dataset*
 - *bbbp_dataset*
 - *chembl_filtered_dataset*
 - *clintox_dataset*
 - *davis_dataset*
 - *ddi_dataset*
 - *dti_dataset*
 - *esol_dataset*
 - *freesolv_dataset*
 - *hiv_dataset*
 - *inmemory_dataset*
 - *kiba_dataset*

- *lipophilicity_dataset*
- *mu_v_dataset*
- *ppi_dataset*
- *sider_dataset*
- *tox21_dataset*
- *toxcast_dataset*
- *zinc_dataset*
- *Helpful Link*

5.5.1 bace_dataset

Processing of bace dataset.

It contains quantitative IC50 and qualitative (binary label) binding results for a set of inhibitors of human beta-secretase 1 (BACE=1). The data are experimental values collected from the scientific literature which contains 152 compounds and their 2D structures and properties

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators

`pahelix.datasets.bace_dataset.get_default_bace_task_names()`

Get that default bace task names.

`pahelix.datasets.bace_dataset.load_bace_dataset(data_path, task_names=None)`

Load bace dataset ,process the classification labels and the input information.

Description:

The data file contains a csv table, in which columns below are used:

- mol: The smile representation of the molecular structure;
- pIC50: The negative log of the IC50 binding affinity;
- class: The binary labels for inhibitor.

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_bace_dataset('./bace')
print(len(dataset))
```

References:

[1]Subramanian, Govindan, et al. “Computational modeling of -secretase 1 (BACE-1) inhibitors using ligand based approaches.” Journal of chemical information and modeling 56.10 (2016): 1936-1949.

5.5.2 bbbp_dataset

Processing of Blood-Brain Barrier Penetration dataset

The Blood-brain barrier penetration (BBBP) dataset is extracted from a study on the modeling and prediction of the barrier permeability. As a membrane separating circulating blood and brain extracellular fluid, the blood-brain barrier blocks most drugs, hormones and neurotransmitters. Thus penetration of the barrier forms a long-standing issue in development of drugs targeting central nervous system. This dataset includes binary labels for over 2000 compounds on their permeability properties.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators

```
pahelix.datasets.bbbp_dataset.get_default_bbbp_task_names()
```

Get that default bbbp task names and return the binary labels

```
pahelix.datasets.bbbp_dataset.load_bbbp_dataset(data_path, task_names=None)
```

Load bbbp dataset ,process the classification labels and the input information.

Description:

The data file contains a csv table, in which columns below are used:

Num:number

name:Name of the compound

smiles:SMILES representation of the molecular structure

p_np:Binary labels for penetration/non-penetration

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_bbbp_dataset('./bbbp')
print(len(dataset))
```

References:

[1] Martins, Ines Filipa, et al. "A Bayesian approach to in silico blood-brain barrier penetration modeling." Journal of chemical information and modeling 52.6 (2012): 1686-1697.

5.5.3 chembl_filtered_dataset

Processing of chembl filtered dataset.

The ChEMBL dataset containing 456K molecules with 1310 kinds of diverse and extensive biochemical assays. The database is unique because of its focus on all aspects of drug discovery and its size, containing information on more than 1.8 million compounds and over 15 million records of their effects on biological systems.

`pahelix.datasets.chembl_filtered_dataset.get_chembl_filtered_task_num()`

Get that default bace task names and return class

`pahelix.datasets.chembl_filtered_dataset.load_chembl_filtered_dataset(data_path)`

Load chembl_filtered dataset ,process the classification labels and the input information.

Introduction:

Note that, in order to load this dataset, you should have other datasets (bace, bbbp, clintox, esol, freesolv, hiv, lipophilicity, muv, sider, tox21, toxcast) downloaded. Since the chembl dataset may overlap with the above listed dataset, the overlapped smiles for test will be filtered for a fair evaluation.

Description:

The data file contains a csv table, in which columns below are used:

It contains the ID, SMILES/CTAB, InChI and InChIKey compound information

smiles: SMILES representation of the molecular structure

Parameters

data_path (*str*) – the path to the cached npz path

Returns

an InMemoryDataset instance.

Example

```
dataset = load_bbbp_dataset('./bace')
print(len(dataset))
```

References:

[1] Gaulton, A; et al. (2011). "ChEMBL: a large-scale bioactivity database for drug discovery". Nucleic Acids Research. 40 (Database issue): D1100-7.

5.5.4 clintox_dataset

Processing of clintox dataset

The ClinTox dataset compares drugs approved by the FDA and drugs that have failed clinical trials for toxicity reasons. The dataset includes two classification tasks for 1491 drug compounds with known chemical structures: (1) clinical trial toxicity (or absence of toxicity) and (2) FDA approval status. List of FDA-approved drugs are compiled from the SWEETLEAD database, and list of drugs that failed clinical trials for toxicity reasons are compiled from the Aggregate Analysis of ClinicalTrials.gov(AACT) database.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators

```
pahelix.datasets.clintox_dataset.get_default_clintox_task_names()
```

Get that default clintox task names and return class

```
pahelix.datasets.clintox_dataset.load_clintox_dataset(data_path, task_names=None)
```

Load Clintox dataset ,process the classification labels and the input information.

Description:

The data file contains a csv table, in which columns below are used:

- smiles: SMILES representation of the molecular structure
- FDA_APPROVED: FDA approval status
- CT_TOX: Clinical trial results

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_clintox_dataset('./clintox')
print(len(dataset))
```

References:

- [1] Gayvert, Kaitlyn M., Neel S. Madhukar, and Olivier Elemento. “A data-driven approach to predicting successes and failures of clinical trials.” Cell chemical biology 23.10 (2016): 1294-1301.
- [2] Artemov, Artem V., et al. “Integrated deep learned transcriptomic and structure-based predictor of clinical trials outcomes.” bioRxiv (2016): 095653.
- [3] Novick, Paul A., et al. “SWEETLEAD: an in silico database of approved drugs, regulated chemicals, and herbal isolates for computer-aided drug discovery.” PloS one 8.11 (2013): e79568.
- [4] Aggregate Analysis of ClinicalTrials.gov (AACT) Database. <https://www.ctti-clinicaltrials.org/aact-database>

5.5.5 davis_dataset

Processing of davis dataset

```
pahelix.datasets.davis_dataset.load_davis_dataset(data_path, featurizer)
    tbd
```

5.5.6 ddi_dataset

Processing of ddi dataset. The DDI dataset includes 23,052 Drug-Drug Synergy pairs from 39 celllines. You can download the dataset from <http://www.bioinf.jku.at/software/DeepSynergy/labels.csv> and load it into pahelix reader creators

```
pahelix.datasets.ddi_dataset.get_default_ddi_task_names()
```

Get that default ddi task names and return class label

```
pahelix.datasets.ddi_dataset.load_ddi_dataset(data_path, task_names=None, cellline=None)
```

Load ddi dataset, process the input information.

Description:

The data file contains a csv table, in which columns below are used:

drug_a_name: drug name

drug_b_name: drug name

cell_line: cell line which the drug pairs were tested on

synergy: continuous values represent the synergy effect, we use 30 as threshold to binarize the data into binary labels. 1 as positive and 0 as negative

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.
- **cellline** – the exact cellline model you want to test on.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_hddi_dataset('./ddi/raw')
print(len(dataset))
```

References:

[1] Drug-Drug Dynergy Data. <https://academic.oup.com/bioinformatics/advance-article/doi/10.1093/bioinformatics/btx806/4747884>

5.5.7 dti_dataset

Processing of DTi dataset. The DTi dataset were extracted from the DrugCombDB. You can download the dataset from http://drugcombdb.denglab.org/download/drug_protein_links.rar and load it into pahelix reader creators

```
pahelix.datasets.dti_dataset.get_default_dti_task_names()
```

Get that default dti task names

```
pahelix.datasets.dti_dataset.load_dti_dataset(data_path, task_names=None, featurizer=None)
```

Load dti dataset, process the input information and the featurizer.

Description:

The data file contains a tsv table, in which columns below are used:

chemical: drug name

protein: targeted protein name

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_hddi_dataset('./dti/raw')
print(len(dataset))
```

5.5.8 esol_dataset

Processing of esol dataset.

ESOL (delaney) is a standard regression data set, which is also called delaney dataset. In the dataset, you can find the structure and water solubility data of 1128 compounds. It's a good choice to validate machine learning models and to estimate solubility directly based on molecular structure which was encoded in SMILES string.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators.

```
pahelix.datasets.esol_dataset.get_default_esol_task_names()
```

Get that default esol task names and return measured values

```
pahelix.datasets.esol_dataset.get_esol_stat(data_path, task_names)
```

Return mean and std of labels

```
pahelix.datasets.esol_dataset.load_esol_dataset(data_path, task_names=None)
```

Load esol dataset, process the classification labels and the input information.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure

Compound ID: Name of the compound

measured log solubility in mols per litre: Log-scale water solubility of the compound, used as label

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_esol_dataset('./esol')
print(len(dataset))
```

References:

[1] Delaney, John S. “ESOL: estimating aqueous solubility directly from molecular structure.” Journal of chemical information and computer sciences 44.3 (2004): 1000-1005.

5.5.9 freesolv_dataset

Processing of freesolv dataset.

The Free Solvation Dataset provides rich information. It contains calculated values and experimental values about hydration free energy of small molecules in water. You can get the calculated values by molecular dynamics simulations, which are derived from alchemical free energy calculations. However, the experimental values are included in the benchmark collection.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators.

```
pahelix.datasets.freesolv_dataset.get_default_freesolv_task_names()
```

Get that default freesolv task names and return measured expt

```
pahelix.datasets.freesolv_dataset.get_freesolv_stat(data_path, task_names)
```

Return mean and std of labels

```
pahelix.datasets.freesolv_dataset.load_freesolv_dataset(data_path, task_names=None)
```

Load freesolv dataset, process the input information and the featurizer.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure

Compound ID: Name of the compound

measured log solubility in mols per litre: Log-scale water solubility of the compound, used as label.

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_freesolv_dataset('./freesolv')
print(len(dataset))
```

References:

[1] Mobley, David L., and J. Peter Guthrie. “FreeSolv: a database of experimental and calculated hydration free energies, with input files.” *Journal of computer-aided molecular design* 28.7 (2014): 711-720.

[2] <https://github.com/MobleyLab/FreeSolv>

5.5.10 hiv_dataset

Processing of hiv dataset.

The HIV dataset was introduced by the Drug Therapeutics Program (DTP) AIDS Antiviral Screen, which tested the ability to inhibit HIV replication for over 40,000 compounds. Screening results were evaluated and placed into three categories: confirmed inactive (CI), confirmed active (CA) and confirmed moderately active (CM). We further combine the latter two labels, making it a classification task between inactive (CI) and active (CA and CM).

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators

```
pahelix.datasets.hiv_dataset.get_default_hiv_task_names()
```

Get that default hiv task names and return class label

```
pahelix.datasets.hiv_dataset.load_hiv_dataset(data_path, task_names=None)
```

Load hiv dataset, process the input information.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure

activity: Three-class labels for screening results: CI/CM/CA.

HIV_active: Binary labels for screening results: 1 (CA/CM) and 0 (CI)

Parameters

- **data_path** (*str*) – the path to the cached npz path
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_hiv_dataset('./hiv')
print(len(dataset))
```

References:

[1] AIDS Antiviral Screen Data. <https://wiki.nci.nih.gov/display/NCIDTPdata/AIDS+Antiviral+Screen+Data>

5.5.11 inmemory_dataset

In-memory dataset.

```
class pahelix.datasets.inmemory_dataset.InMemoryDataset(data_list=None, npz_data_path=None,
                                                         npz_data_files=None)
```

Description:

The InMemoryDataset manages `data_list` which is a list of *data* and the *data* is a dict of numpy ndarray. And each dict has the same keys.

It works like a list: you can call `dataset[i]` to get the *i*-th element of the `data_list` and call `len(dataset)` to get the length of `data_list`.

The `data_list` can be cached in npz files by calling `dataset.save_data(data_path)` and after that, call `InMemoryDataset(data_path)` to reload.

data_list

a list of dict of numpy ndarray.

Type

list

Example

```
data_list = [{'a': np.zeros([4, 5])}, {'a': np.zeros([7, 5])}]
dataset = InMemoryDataset(data_list=data_list)
print(len(dataset))
dataset.save_data('./cached_npz')    # save data_list to ./cached_npz

dataset2 = InMemoryDataset(npz_data_path='./cached_npz')    # will load the saved
↪ `data_list`
print(len(dataset))
```

```
get_data_loader(batch_size, num_workers=4, shuffle=False, collate_fn=None)
```

It returns an batch iterator which yields a batch of data. Firstly, a sub-list of *data* of size `batch_size` will be draw from the `data_list`, then the function `collate_fn` will be applied to the sub-list to create a batch and yield back. This process is accelerated by multiprocessing.

Parameters

- **batch_size** (*int*) – the batch_size of the batch data of each yield.
- **num_workers** (*int*) – the number of workers used to generate batch data. Required by multiprocessing.
- **shuffle** (*bool*) – whether to shuffle the order of the `data_list`.

- **collate_fn** (*function*) – used to convert the sub-list of `data_list` to the aggregated batch data.

Yields

the batch data processed by `collate_fn`.

save_data(*data_path*)

Save the `data_list` to the disk specified by `data_path` with npz format. After that, call `InMemoryDataset(data_path)` to reload the `data_list`.

Parameters

data_path (*str*) – the path to the cached npz path.

transform(*transform_fn*, *num_workers=4*, *drop_none=False*)

Inplace apply *transform_fn* on the `data_list` with multiprocessing.

5.5.12 kiba_dataset

Processing of kiba dataset

```
pahelix.datasets.kiba_dataset.load_kiba_dataset(data_path, featurizer)
tbd
```

5.5.13 lipophilicity_dataset

Processing of lipophilicity dataset.

Lipophilicity is a dataset curated from ChEMBL database containing experimental results on octanol/water distribution coefficient (logD at pH=7.4). As the Lipophilicity plays an important role in membrane permeability and solubility. Related work deserves more attention.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators.

```
pahelix.datasets.lipophilicity_dataset.get_default_lipophilicity_task_names()
```

Get that default lipophilicity task names and return measured expt

```
pahelix.datasets.lipophilicity_dataset.get_lipophilicity_stat(data_path, task_names)
```

Return mean and std of labels

```
pahelix.datasets.lipophilicity_dataset.load_lipophilicity_dataset(data_path,
                                                                    task_names=None)
```

Load lipophilicity dataset, process the input information.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure

exp: Measured octanol/water distribution coefficient (logD) of the compound, used as label

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an `InMemoryDataset` instance.

Example

```
dataset = load_lipophilicity_dataset('./lipophilicity')
print(len(dataset))
```

References:

[1]Hersey, A. ChEMBL Deposited Data Set - AZ dataset; 2015. <https://doi.org/10.6019/chembl3301361>

5.5.14 muv_dataset

Processing of muv dataset.

The Maximum Unbiased Validation (MUV) group is a benchmark dataset selected from PubChem BioAssay by applying a refined nearest neighbor analysis. The MUV dataset contains 17 challenging tasks for around 90,000 compounds and is specifically designed for validation of virtual screening techniques.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators.

`pahelix.datasets.muv_dataset.get_default_muv_task_names()`

Get that default hiv task names and return the measured results for bioassays

`pahelix.datasets.muv_dataset.load_muv_dataset(data_path, task_names=None)`

Load muv dataset,process the input information.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure.

mol_id: PubChem CID of the compound.

MUV-XXX: Measured results (Active/Inactive) for bioassays.

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_muv_dataset('./muv')
print(len(dataset))
```

References:

[1]Rohrer, Sebastian G., and Knut Baumann. “Maximum unbiased validation (MUV) data sets for virtual screening based on PubChem bioactivity data.” Journal of chemical information and modeling 49.2 (2009): 169-184.

5.5.15 ppi_dataset

Processing of PPI dataset. The DDI dataset were extracted from DrugCombDB. You can download the dataset from http://drugcombdb.denglab.org/download/protein_protein_links.rar and load it into pahelix reader creators

```
pahelix.datasets.ppi_dataset.get_default_ppi_task_names()
```

Get that default ppi task names

```
pahelix.datasets.ppi_dataset.load_ppi_dataset(data_path, task_names=None, featurizer=None)
```

Load ppi dataset, process the input information and the featurizer.

Description:

The data file contains a txt file, in which columns below are used:

protein1: protein1 name

protein2: protein2 name

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the txt file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_ppi_dataset('./ppi/raw')
print(len(dataset))
```

5.5.16 sider_dataset

Processing of sider dataset.

The Side Effect Resource (SIDER) is a database of marketed drugs and adverse drug reactions (ADR). The version of the SIDER dataset in DeepChem has grouped drug side effects into 27 system organ classes following MedDRA classifications measured for 1427 approved drugs.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators.

```
pahelix.datasets.sider_dataset.get_default_sider_task_names()
```

Get that default sider task names and return the side results for the drug

```
pahelix.datasets.sider_dataset.load_sider_dataset(data_path, task_names=None)
```

Load sider dataset, process the input information.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure.

Hepatobiliary disorders: Injury, poisoning and procedural complications, recorded side effects for the drug

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_sider_dataset('./sider')
print(len(dataset))
```

References:

[1]Kuhn, Michael, et al. “The SIDER database of drugs and side effects.” Nucleic acids research 44.D1 (2015): D1075-D1079.

[2]Altae-Tran, Han, et al. “Low data drug discovery with one-shot learning.” ACS central science 3.4 (2017): 283-293.

[3]Medical Dictionary for Regulatory Activities. <http://www.meddra.org/>

[4]Please refer to <http://sideeffects.embl.de/se/?page=98> for details on ADRs.

5.5.17 tox21_dataset

Processing of tox21 dataset.

The “Toxicology in the 21st Century” (Tox21) initiative created a public database measuring toxicity of compounds, which has been used in the 2014 Tox21 Data Challenge. This dataset contains qualitative toxicity measurements for 8k compounds on 12 different targets, including nuclear receptors and stress response pathways.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators.

`pahelix.datasets.tox21_dataset.get_default_tox21_task_names()`

Get that default tox21 task names and return the bioassays results

`pahelix.datasets.tox21_dataset.load_tox21_dataset(data_path, task_names=None)`

Load tox21 dataset, process the input information.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure.

NR-XXX: Nuclear receptor signaling bioassays results.

SR-XXX: Stress response bioassays results

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_tox21_dataset('./tox21')
print(len(dataset))
```

References:

[1]Tox21 Challenge. <https://tripod.nih.gov/tox21/challenge/>

[2]please refer to the links at <https://tripod.nih.gov/tox21/challenge/data.jsp> for details.

5.5.18 toxcast_dataset

Processing of toxcast dataset.

ToxCast is an extended data collection from the same initiative as Tox21, providing toxicology data for a large library of compounds based on in vitro high-throughput screening. The processed collection includes qualitative results of over 600 experiments on 8k compounds.

You can download the dataset from <http://moleculenet.ai/datasets-1> and load it into pahelix reader creators.

`pahelix.datasets.toxcast_dataset.get_default_toxcast_task_names(data_path)`

Get that default toxcast task names and return the list of the input information

`pahelix.datasets.toxcast_dataset.load_toxcast_dataset(data_path, task_names=None)`

Load toxcast dataset,process the input information.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure.

ACEA_T47D_80hr_Negative: “Tanguay_ZF_120hpf_YSE_up” - Bioassays results

SR-XXX: Stress response bioassays results

Parameters

- **data_path** (*str*) – the path to the cached npz path.
- **task_names** (*list*) – a list of header names to specify the columns to fetch from the csv file.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_toxcast_dataset('./toxcast')
print(len(dataset))
```

References:

[1]Richard, Ann M., et al. “ToxCast chemical landscape: paving the road to 21st century toxicology.” Chemical research in toxicology 29.8 (2016): 1225-1251.

[2]please refer to the section “high-throughput assay information” at <https://www.epa.gov/chemical-research/toxicity-forecaster-toxcasttm-data> for details.

5.5.19 zinc_dataset

Processing of ZINC dataset.

The ZINC database is a curated collection of commercially available chemical compounds prepared especially for virtual screening. ZINC15 is designed to bring together biology and chemoinformatics with a tool that is easy to use for nonexperts, while remaining fully programmable for chemoinformaticians and computational biologists.

`pahelix.datasets.zinc_dataset.load_zinc_dataset(data_path)`

Load ZINC dataset, process the input information.

Description:

The data file contains a csv table, in which columns below are used:

smiles: SMILES representation of the molecular structure.

zinc_id: the id of the compound

Parameters

data_path (*str*) – the path to the cached npz path.

Returns

an InMemoryDataset instance.

Example

```
dataset = load_zinc_dataset('./zinc')
print(len(dataset))
```

References:

[1]Teague Sterling and John J. Irwin. Zinc 15 – ligand discovery for everyone. Journal of Chemical Information and Modeling, 55(11):2324–2337, 2015. doi: 10.1021/acs.jcim.5b00559. PMID: 26479676.

5.5.20 Helpful Link

Please refer to our [GitHub repo](#) to see the whole module.

5.6 pahelix.featurizers

Table of Contents

- *pahelix.featurizers*
 - *het_gnn_featurizer*
 - *pretrain_gnn_featurizer*
 - *Helpful Link*

5.6.1 het_gnn_featurizer

Featurizers for DDI Heterogenous graph.

```
class pahelix.featurizers.het_gnn_featurizer.DDiFeaturizer
```

Featurizer for drugs

```
collate_fn(ddi_data, dti_data, ppi_data, features)
```

Aggregate all needed nodes into a Hetrogenous graph

```
pahelix.featurizers.het_gnn_featurizer.num_nodes_stat(data)
```

count the number of nodes from data

Examples

```
data: {'pair': (a, b)}
```

```
pahelix.featurizers.het_gnn_featurizer.nx_graph_build(hg, nodes_dict, label)
```

Build Heterogenous graph with node name not idx.

5.6.2 pretrain_gnn_featurizer

Featurizers for pretrain-gnn.

Adapted from <https://github.com/snap-stanford/pretrain-gnns/tree/master/chem/utils.py>

```
class pahelix.featurizers.pretrain_gnn_featurizer.AttrmaskTransformFn
```

Gen features for attribute mask model of pretrain gnns

```
class pahelix.featurizers.pretrain_gnn_featurizer.AttrmaskCollateFn(atom_names, bond_names,  
                                                                    mask_ratio=0.15)
```

CollateFn for attribute mask model of pretrain gnns

```
class pahelix.featurizers.pretrain_gnn_featurizer.SupervisedTransformFn
```

Gen features for supervised model of pretrain gnns

```
class pahelix.featurizers.pretrain_gnn_featurizer.SupervisedCollateFn(atom_names,  
                                                                    bond_names)
```

CollateFn for supervised model of pretrain gnns

5.6.3 Helpful Link

Please refer to our [GitHub repo](#) to see the whole module.

5.7 pahelix.model_zoo

Table of Contents

- *pahelix.model_zoo*
 - *pretrain_gnns_model*
 - *protein_sequence_model*
 - *seq_vae_model*
 - *Helpful Link*

5.7.1 pretrain_gnns_model

This is an implementation of pretrain gnns: <https://arxiv.org/abs/1905.12265>

class `pahelix.model_zoo.pretrain_gnns_model.AttrMaskModel(*args: Any, **kwargs: Any)`

This is a pretraining model used by pretrain gnns for attribute mask training.

Returns

the loss variance of the model.

Return type

loss

forward(*graphs*, *masked_node_indice*, *masked_node_labels*)

Build the network.

class `pahelix.model_zoo.pretrain_gnns_model.PretrainGNNModel(*args: Any, **kwargs: Any)`

The basic GNN Model used in pretrain gnns.

Parameters

model_config (*dict*) – a dict of model configurations.

forward(*graph*)

Build the network.

property graph_dim

the out dim of graph_repr

property node_dim

the out dim of graph_repr

class `pahelix.model_zoo.pretrain_gnns_model.SupervisedModel(*args: Any, **kwargs: Any)`

This is a pretraining model used by pretrain gnns for supervised training.

Returns

the loss variance of the model.

Return type

self.loss

forward(*graphs*, *labels*, *valids*)

Build the network.

5.7.2 protein_sequence_model

Sequence-based models for protein.

```
class pahelix.model_zoo.protein_sequence_model.LstmEncoderModel(vocab_size, emb_dim=128,  
                                                                hidden_size=1024, n_layers=3,  
                                                                padding_idx=0, epsilon=1e-05,  
                                                                dropout_rate=0.1)
```

```
    forward(input, pos)
```

```
class pahelix.model_zoo.protein_sequence_model.ResnetEncoderModel(vocab_size, emb_dim=128,  
                                                                hidden_size=256,  
                                                                kernel_size=9, n_layers=35,  
                                                                padding_idx=0,  
                                                                dropout_rate=0.1,  
                                                                epsilon=1e-06)
```

```
    forward(input, pos)
```

```
    init_weights(layer)
```

```
        Initialization hook
```

```
class pahelix.model_zoo.protein_sequence_model.TransformerEncoderModel(vocab_size,  
                                                                        emb_dim=512,  
                                                                        hidden_size=512,  
                                                                        n_layers=8,  
                                                                        n_heads=8,  
                                                                        padding_idx=0,  
                                                                        dropout_rate=0.1)
```

```
    forward(input, pos)
```

```
    init_weights(layer)
```

```
        Initialization hook
```

```
class pahelix.model_zoo.protein_sequence_model.PretrainTaskModel(class_num, model_config,  
                                                                encoder_model)
```

```
    forward(input, pos)
```

```
class pahelix.model_zoo.protein_sequence_model.SeqClassificationTaskModel(class_num,  
                                                                           model_config,  
                                                                           encoder_model)
```

```
    forward(input, pos)
```

```
class pahelix.model_zoo.protein_sequence_model.ClassificationTaskModel(class_num,  
                                                                        model_config,  
                                                                        encoder_model)
```

```
    forward(input, pos)
```

```
class pahelix.model_zoo.protein_sequence_model.RegressionTaskModel(model_config,  
                                                                    encoder_model)
```

```
    forward(input, pos)
```

```

class pahelix.model_zoo.protein_sequence_model.ProteinEncoderModel(model_config, name='')
    ProteinSequenceModel
    forward(input, pos)

class pahelix.model_zoo.protein_sequence_model.ProteinModel(encoder_model, model_config)
    forward(input, pos)

class pahelix.model_zoo.protein_sequence_model.ProteinCriterion(model_config)
    cal_loss(pred, label)

```

5.7.3 seq_vae_model

```

class pahelix.model_zoo.seq_vae_model.VAE(vocab, model_config)
    The sequence VAE model

    Parameters
        • vocab – the vocab object.
        • model_config – the json files of model parameters.

    forward(x)
        Model forward

    forward_decoder(x, z)
        decoder

    forward_encoder(x)
        encoder

    sample(n_batch, max_len=100, z=None, temp=1.0)
        Generating n_batch samples in eval mode (z could be not on same device)

        Parameters
            • n_batch – number of sentences to generate
            • max_len – max len of samples
            • z – (n_batch, d_z) of floats, latent vector z or None
            • temp – temperature of softmax

        Returns
            list of tensors of strings, samples sequence x

    sample_z_prior(n_batch)
        Sampling  $z \sim p(z) = N(0, I)$ 

        Parameters
            • n_batch – number of batches

        Returns
            (n_batch, d_z) of floats, sample of latent z

    tensor2string(tensor)
        convert tensor values to sequence string

```

5.7.4 Helpful Link

Please refer to our [GitHub repo](#) to see the whole module.

5.8 pahelix.networks

Table of Contents

- *pahelix.networks*
 - *basic_block*
 - *compound_encoder*
 - *gnn_block*
 - *involution_block*
 - *lstm_block*
 - *optimizer*
 - *pre_post_process*
 - *resnet_block*
 - *transformer_block*
 - *Helpful Link*

5.8.1 basic_block

Some frequently used basic blocks

```
class pahelix.networks.basic_block.Activation(*args: Any, **kwargs: Any)
```

```
    forward(x)  
        tbd
```

```
class pahelix.networks.basic_block.MLP(*args: Any, **kwargs: Any)
```

```
    forward(x)
```

Parameters

\mathbf{x} (tensor) – (-1, dim).

5.8.2 compound_encoder

Basic Encoder for compound atom/bond features.

```
class pahelix.networks.compound_encoder.AtomEmbedding(*args: Any, **kwargs: Any)
```

Atom Encoder

```
forward(node_features)
```

Parameters

node_features (*dict of tensor*) – node features.

```
class pahelix.networks.compound_encoder.AtomFloatEmbedding(*args: Any, **kwargs: Any)
```

Atom Float Encoder

```
forward(feats)
```

Parameters

feats (*dict of tensor*) – node float features.

```
class pahelix.networks.compound_encoder.BondAngleFloatRBF(*args: Any, **kwargs: Any)
```

Bond Angle Float Encoder using Radial Basis Functions

```
forward(bond_angle_float_features)
```

Parameters

bond_angle_float_features (*dict of tensor*) – bond angle float features.

```
class pahelix.networks.compound_encoder.BondEmbedding(*args: Any, **kwargs: Any)
```

Bond Encoder

```
forward(edge_features)
```

Parameters

edge_features (*dict of tensor*) – edge features.

```
class pahelix.networks.compound_encoder.BondFloatRBF(*args: Any, **kwargs: Any)
```

Bond Float Encoder using Radial Basis Functions

```
forward(bond_float_features)
```

Parameters

bond_float_features (*dict of tensor*) – bond float features.

5.8.3 gnn_block

Blocks for Graph Neural Network (GNN)

Adapted from <https://github.com/snap-stanford/pretrain-gnns/blob/master/chem/model.py>

```
class pahelix.networks.gnn_block.GIN(*args: Any, **kwargs: Any)
```

Implementation of Graph Isomorphism Network (GIN) layer with edge features

```
forward(graph, node_feat, edge_feat)
```

Parameters

- **node_feat** (*tensor*) – node features with shape (num_nodes, feature_size).
- **edge_feat** (*tensor*) – edges features with shape (num_edges, feature_size).

class pahelix.networks.gnn_block.**GraphNorm**(*args: Any, **kwargs: Any)

Implementation of graph normalization. Each node features is divided by $\sqrt{\text{num_nodes}}$ per graphs.

Parameters

- **graph** – the graph object from (Graph)
- **feature** – A tensor with shape (num_nodes, feature_size).

Returns

A tensor with shape (num_nodes, hidden_size)

References:

[1] BENCHMARKING GRAPH NEURAL NETWORKS. <https://arxiv.org/abs/2003.00982>

forward(graph, feature)

graph norm

class pahelix.networks.gnn_block.**MeanPool**(*args: Any, **kwargs: Any)

TODO: temporary class due to pgl mean pooling

forward(graph, node_feat)

mean pooling

5.8.4 involution_block

class pahelix.networks.involution_block.**Involution2D**(in_channel, out_channel,
sigma_mapping=None, kernel_size=7,
stride=1, groups=1, reduce_ratio=1,
dilation=1, padding=3)

Involution module.

Parameters

- **in_channel** – The channel size of input.
- **out_channel** – The channel size of output.
- **sigma_mapping** – Sigma mapping.
- **kernel_size** – Kernel size.
- **stride** – Stride size.
- **groups** – Group size.
- **reduce_ratio** – The ratio of reduce.
- **dilation** – The dilation size.
- **padding** – The padding size.

Returns

The output of Involution2D block.

Return type

output

References:

[1] Involution: Inverting the Inference of Convolution for Visual Recognition. <https://arxiv.org/abs/2103.06255>

forward(*x*)

Involution block

5.8.5 lstm_block

Lstm block.

`pahelix.networks.lstm_block.lstm_encoder(input, hidden_size, n_layer=1, is_bidirectory=True, param_initializer=None, name='lstm')`

The encoder is composed of a stack of lstm layers.

Parameters

- **input** – The input of lstm encoder.
- **hidden_size** – The hidden size of lstm.
- **n_layer** – The number of lstm layers.
- **is_bidirectory** – True if the lstm is bidirectory.
- **param_initializer** – The parameter initializer for lstm encoder.
- **name** – The prefix of the parameters' name in lstm encoder.

Returns

The hidden units of lstm encoder. checkpoints: The checkpoints for recompute mechanism.

Return type

hidden

5.8.6 optimizer

`class pahelix.networks.optimizer.AdamW(*args, **kwargs)`

AdamW object for dygraph.

`apply_optimize(loss, startup_program, params_grads)`

Update params with weight decay.

5.8.7 pre_post_process

`pahelix.networks.pre_post_process.pre_post_process_layer(prev_out, out, process_cmd, dropout_rate=0.0, epsilon=1e-05, name='', is_test=False)`

Add residual connection, layer normalization and dropout to the out tensor optionally according to the value of process_cmd.

This will be used before or after multi-head attention and position-wise feed-forward networks.

5.8.8 resnet_block

Resnet block.

```
pahelix.networks.resnet_block.resnet_encoder(input, hidden_size, n_layer=1, filter_size=3, act='gelu',  
                                              epsilon=1e-06, param_initializer=None, name='resnet')
```

The encoder is composed of a stack of resnet layers.

Parameters

- **input** – The input of resnet encoder.
- **hidden_size** – The hidden size of resnet.
- **n_layer** – The number of resnet layers.
- **act** – The activation function.
- **param_initializer** – The parameter initializer for resnet encoder.
- **name** – The prefix of the parameters' name in resnet encoder.

Returns

The hidden units of resnet encoder. checkpoints: The checkpoints for recompute mechanism.

Return type

hidden

5.8.9 transformer_block

Transformer block.

```
pahelix.networks.transformer_block.multi_head_attention(queries, keys, values, attn_bias, d_key,  
                                                         d_value, d_model, n_head=1,  
                                                         dropout_rate=0.0, cache=None,  
                                                         gather_idx=None, store=False,  
                                                         param_initializer=None, lr=1.0,  
                                                         name='multi_head_att', is_test=False)
```

Multi-Head Attention.

Note that `attn_bias` is added to the logit before computing softmax activation to mask certain selected positions so that they will not be considered in attention weights.

```
pahelix.networks.transformer_block.positionwise_feed_forward(x, d_inner_hid, d_hid, dropout_rate,  
                                                             hidden_act, num_flatten_dims=2,  
                                                             param_initializer=None, name='ffn',  
                                                             is_test=False)
```

Position-wise Feed-Forward Networks.

This module consists of two linear transformations with a ReLU activation in between, which is applied to each position separately and identically.

```
pahelix.networks.transformer_block.transformer_encoder(enc_input, attn_bias, n_layer, n_head,
                                                       d_key, d_value, d_model, d_inner_hid,
                                                       prepostprocess_dropout, attention_dropout,
                                                       act_dropout, hidden_act,
                                                       preprocess_cmd='n', postprocess_cmd='da',
                                                       param_initializer=None, name="",
                                                       epsilon=1e-05, n_layer_per_block=1,
                                                       param_share='normal', caches=None,
                                                       gather_idx=None, store=False,
                                                       is_test=False)
```

The encoder is composed of a stack of identical layers returned by calling `transformer_encoder_layer`.

```
pahelix.networks.transformer_block.transformer_encoder_layer(input, attn_bias, n_head, d_key,
                                                             d_value, d_model, d_inner_hid,
                                                             prepostprocess_dropout,
                                                             attention_dropout, act_dropout,
                                                             hidden_act, preprocess_cmd='n',
                                                             postprocess_cmd='da',
                                                             param_initializer=None, name="",
                                                             epsilon=1e-05, cache=None,
                                                             gather_idx=None, store=False,
                                                             is_test=False)
```

The encoder layers that can be stacked to form a deep encoder.

This module consists of a multi-head (self) attention followed by position-wise feed-forward networks and both the two components accompanied with the `pre_process_layer` / `post_process_layer` to add residual connection, layer normalization and dropout.

5.8.10 Helpful Link

Please refer to our [GitHub repo](#) to see the whole module.

5.9 pahelix.utils

Table of Contents

- *pahelix.utils*
 - *basic_utils*
 - *compound_tools*
 - *data_utils*
 - *language_model_tools*
 - *protein_tools*
 - *splitters*
 - *Helpful Link*

5.9.1 basic_utils

Basic utils

```
pahelix.utils.basic_utils.load_json_config(path)
    tbd

pahelix.utils.basic_utils.mp_pool_map(list_input, func, num_workers)
    list_output = [func(input) for input in list_input]
```

5.9.2 compound_tools

Tools for compound features.

Adapted from <https://github.com/snap-stanford/pretrain-gnns/blob/master/chem/loader.py>

```
class pahelix.utils.compound_tools.Compound3DKit
    the 3Dkit of Compound

    static get_2d_atom_poses(mol)
        get 2d atom poses

    static get_MMFF_atom_poses(mol, numConfs=None, return_energy=False)
        the atoms of mol will be changed in some cases.

    static get_atom_poses(mol, conf)
        tbd

    static get_bond_lengths(edges, atom_poses)
        get bond lengths

    static get_superedge_angles(edges, atom_poses, dir_type='HT')
        get superedge angles

class pahelix.utils.compound_tools.CompoundKit

    static atom_to_feat_vector(atom)
        tbd

    static check_partial_charge(atom)
        tbd

    static get_atom_feature_id(atom, name)
        get atom features id

    static get_atom_feature_size(name)
        get atom features size

    static get_atom_names(mol)
        get atom name list TODO: to be remove in the future

    static get_atom_value(atom, name)
        get atom values
```

```

static get_bond_feature_id(bond, name)
    get bond features id
static get_bond_feature_size(name)
    get bond features size
static get_bond_value(bond, name)
    get bond values
static get_daylight_functional_group_counts(mol)
    get daylight functional group counts
static get_maccs_fingerprint(mol)
    get maccs fingerprint
static get_morgan2048_fingerprint(mol, radius=2)
    get morgan2048 fingerprint
static get_morgan_fingerprint(mol, radius=2)
    get morgan fingerprint
static get_ring_size(mol)
    return (N,6) list

```

`pahelix.utils.compound_tools.check_smiles_validity(smiles)`

Check whether the smile can't be converted to rdkit mol object.

`pahelix.utils.compound_tools.create_standardized_mol_id(smiles)`

Parameters

smiles – smiles sequence.

Returns

inchi.

`pahelix.utils.compound_tools.get_atom_feature_dims(list_acquired_feature_names)`

tbd

`pahelix.utils.compound_tools.get_bond_feature_dims(list_acquired_feature_names)`

tbd

`pahelix.utils.compound_tools.get_gasteiger_partial_charges(mol, n_iter=12)`

Calculates list of gasteiger partial charges for each atom in mol object.

Parameters

- **mol** – rdkit mol object.
- **n_iter** (*int*) – number of iterations. Default 12.

Returns

list of computed partial charges for each atom.

`pahelix.utils.compound_tools.get_largest_mol(mol_list)`

Given a list of rdkit mol objects, returns mol object containing the largest num of atoms. If multiple containing largest num of atoms, picks the first one.

Parameters

mol_list (*list*) – a list of rdkit mol object.

Returns

the largest mol.

`pahelix.utils.compound_tools.mol_to_geognn_graph_data(mol, atom_poses, dir_type)`

mol: rdkit molecule dir_type: direction type for bond_angle graph

`pahelix.utils.compound_tools.mol_to_geognn_graph_data_MMFF3d(mol)`

tbd

`pahelix.utils.compound_tools.mol_to_geognn_graph_data_raw3d(mol)`

tbd

`pahelix.utils.compound_tools.mol_to_graph_data(mol)`

Parameters

- **atom_features** – Atom features.
- **edge_features** – Edge features.
- **morgan_fingerprint** – Morgan fingerprint.
- **functional_groups** – Functional groups.

`pahelix.utils.compound_tools.new_mol_to_graph_data(mol)`

mol_to_graph_data

Parameters

- **atom_features** – Atom features.
- **edge_features** – Edge features.
- **morgan_fingerprint** – Morgan fingerprint.
- **functional_groups** – Functional groups.

`pahelix.utils.compound_tools.new_smiles_to_graph_data(smiles, **kwargs)`

Convert smiles to graph data.

`pahelix.utils.compound_tools.rdchem_enum_to_list(values)`

values = {0: rdkit.Chem.rdchem.ChiralType.CHI_UNSPECIFIED, 1: rdkit.Chem.rdchem.ChiralType.CHI_TETRAHEDRAL_CW, 2: rdkit.Chem.rdchem.ChiralType.CHI_TETRAHEDRAL_CCW, 3: rdkit.Chem.rdchem.ChiralType.CHI_OTHER}

`pahelix.utils.compound_tools.safe_index(alist, elem)`

Return index of element e in list l. If e is not present, return the last index

`pahelix.utils.compound_tools.split_rdkit_mol_obj(mol)`

Split rdkit mol object containing multiple species or one species into a list of mol objects or a list containing a single object respectively.

Parameters

mol – rdkit mol object.

5.9.3 data_utils

Tools for data.

`pahelix.utils.data_utils.get_part_files(data_path, trainer_id, trainer_num)`

Split the files in data_path so that each trainer can train from different examples.

`pahelix.utils.data_utils.load_npz_to_data_list(npz_file)`

Reload the data list save by save_data_list_to_npz.

Parameters

npz_file (*str*) – the npz file location.

Returns

a list of data where each data is a dict of numpy ndarray.

`pahelix.utils.data_utils.save_data_list_to_npz(data_list, npz_file)`

Save a list of data to the npz file. Each data is a dict of numpy ndarray.

Parameters

- **data_list** (*list*) – a list of data.
- **npz_file** (*str*) – the npz file location.

5.9.4 language_model_tools

Tools for language models.

`pahelix.utils.language_model_tools.apply_bert_mask(inputs, pad_mask, tokenizer)`

Apply BERT mask to the token_ids.

Parameters

token_ids – The list of token ids.

Returns

The list of masked token ids. labels: The labels for training BERT.

Return type

masked_token_ids

5.9.5 protein_tools

`class pahelix.utils.protein_tools.ProteinTokenizer`

Protein Tokenizer.

convert_token_to_id(*token*)

Converts a token to an id.

Parameters

token – Token.

Returns

The id of the input token.

Return type

id

convert_tokens_to_ids(*tokens*)

Convert multiple tokens to ids.

Parameters**tokens** – The list of tokens.**Returns**

The id list of the input tokens.

Return type

ids

gen_token_ids(*sequence*)

Generate the list of token ids according the input sequence.

Parameters**sequence** – Sequence to be tokenized.**Returns**

The list of token ids.

Return type

token_ids

tokenize(*sequence*)

Split the sequence into token list.

Parameters**sequence** – The sequence to be tokenized.**Returns**

The token lists.

Return type

tokens

5.9.6 splitters

Splitters

class pahelix.utils.splitters.**RandomSplitter**

Random splitter.

split(*dataset*, *frac_train=None*, *frac_valid=None*, *frac_test=None*, *seed=None*)**Parameters**

- **dataset** ([InMemoryDataset](#)) – the dataset to split.
- **frac_train** (*float*) – the fraction of data to be used for the train split.
- **frac_valid** (*float*) – the fraction of data to be used for the valid split.
- **frac_test** (*float*) – the fraction of data to be used for the test split.
- **seed** (*int* / *None*) – the random seed.

class pahelix.utils.splitters.IndexSplitter

Split datasets that has already been ordered. The first *frac_train* proportion is used for train set, the next *frac_valid* for valid set and the final *frac_test* for test set.

split(dataset, frac_train=None, frac_valid=None, frac_test=None)

Parameters

- **dataset** ([InMemoryDataset](#)) – the dataset to split.
- **frac_train** (*float*) – the fraction of data to be used for the train split.
- **frac_valid** (*float*) – the fraction of data to be used for the valid split.
- **frac_test** (*float*) – the fraction of data to be used for the test split.

class pahelix.utils.splitters.ScaffoldSplitter

Adapted from <https://github.com/deepchem/deepchem/blob/master/deepchem/splits/splitters.py>

Split dataset by Bemis-Murcko scaffolds

split(dataset, frac_train=None, frac_valid=None, frac_test=None)

Parameters

- **dataset** ([InMemoryDataset](#)) – the dataset to split. Make sure each element in the dataset has key “smiles” which will be used to calculate the scaffold.
- **frac_train** (*float*) – the fraction of data to be used for the train split.
- **frac_valid** (*float*) – the fraction of data to be used for the valid split.
- **frac_test** (*float*) – the fraction of data to be used for the test split.

class pahelix.utils.splitters.RandomScaffoldSplitter

Adapted from https://github.com/pfnet-research/chainer-chemistry/blob/master/chainer_chemistry/dataset/splitters/scaffold_splitter.py

Split dataset by Bemis-Murcko scaffolds

split(dataset, frac_train=None, frac_valid=None, frac_test=None, seed=None)

Parameters

- **dataset** ([InMemoryDataset](#)) – the dataset to split. Make sure each element in the dataset has key “smiles” which will be used to calculate the scaffold.
- **frac_train** (*float*) – the fraction of data to be used for the train split.
- **frac_valid** (*float*) – the fraction of data to be used for the valid split.
- **frac_test** (*float*) – the fraction of data to be used for the test split.
- **seed** (*int* / *None*) – the random seed.

pahelix.utils.splitters.generate_scaffold(smiles, include_chirality=False)

Obtain Bemis-Murcko scaffold from smiles

Parameters

- **smiles** – smiles sequence
- **include_chirality** – Default=False

Returns

the scaffold of the given smiles.

5.9.7 Helpful Link

Please refer to our [GitHub repo](#) to see the whole module.

PYTHON MODULE INDEX

p

`pahelix.datasets.bace_dataset`, 17
`pahelix.datasets.bbbp_dataset`, 18
`pahelix.datasets.chembl_filtered_dataset`, 19
`pahelix.datasets.clintox_dataset`, 20
`pahelix.datasets.davis_dataset`, 21
`pahelix.datasets.ddi_dataset`, 21
`pahelix.datasets.dti_dataset`, 22
`pahelix.datasets.esol_dataset`, 22
`pahelix.datasets.freesolv_dataset`, 23
`pahelix.datasets.hiv_dataset`, 24
`pahelix.datasets.inmemory_dataset`, 25
`pahelix.datasets.kiba_dataset`, 26
`pahelix.datasets.lipophilicity_dataset`, 26
`pahelix.datasets.muv_dataset`, 27
`pahelix.datasets.ppi_dataset`, 28
`pahelix.datasets.sider_dataset`, 28
`pahelix.datasets.tox21_dataset`, 29
`pahelix.datasets.toxcast_dataset`, 30
`pahelix.datasets.zinc_dataset`, 31
`pahelix.featurizers.het_gnn_featurizer`, 32
`pahelix.featurizers.pretrain_gnn_featurizer`,
32
`pahelix.model_zoo.pretrain_gnns_model`, 33
`pahelix.model_zoo.protein_sequence_model`, 34
`pahelix.networks.basic_block`, 36
`pahelix.networks.compound_encoder`, 37
`pahelix.networks.gnn_block`, 37
`pahelix.networks.lstm_block`, 39
`pahelix.networks.resnet_block`, 40
`pahelix.networks.transformer_block`, 40
`pahelix.utils.basic_utils`, 42
`pahelix.utils.compound_tools`, 42
`pahelix.utils.data_utils`, 45
`pahelix.utils.language_model_tools`, 45
`pahelix.utils.splitters`, 46

INDEX

A

Activation (class in *pahelix.networks.basic_block*), 36
AdamW (class in *pahelix.networks.optimizer*), 39
apply_bert_mask() (in module *pahelix.utils.language_model_tools*), 45
apply_optimize() (in *pahelix.networks.optimizer.AdamW* method), 39
atom_to_feat_vector() (in *pahelix.utils.compound_tools.CompoundKit* static method), 42
AtomEmbedding (class in *pahelix.networks.compound_encoder*), 37
AtomFloatEmbedding (class in *pahelix.networks.compound_encoder*), 37
AttrmaskCollateFn (class in *pahelix.featurizers.pretrain_gnn_featurizer*), 32
AttrmaskModel (class in *pahelix.model_zoo.pretrain_gnns_model*), 33
AttrmaskTransformFn (class in *pahelix.featurizers.pretrain_gnn_featurizer*), 32

B

BondAngleFloatRBF (class in *pahelix.networks.compound_encoder*), 37
BondEmbedding (class in *pahelix.networks.compound_encoder*), 37
BondFloatRBF (class in *pahelix.networks.compound_encoder*), 37

C

cal_loss() (in *pahelix.model_zoo.protein_sequence_model.ProteinCriterion* method), 35
check_partial_charge() (in *pahelix.utils.compound_tools.CompoundKit* static method), 42
check_smiles_validity() (in module *pahelix.utils.compound_tools*), 43
ClassificationTaskModel (class in *pahelix.model_zoo.protein_sequence_model*), 34

collate_fn() (in *pahelix.featurizers.het_gnn_featurizer.DDiFeaturizer* method), 32
Compound3DKit (class in *pahelix.utils.compound_tools*), 42
CompoundKit (class in *pahelix.utils.compound_tools*), 42
convert_token_to_id() (in *pahelix.utils.protein_tools.ProteinTokenizer* method), 45
convert_tokens_to_ids() (in *pahelix.utils.protein_tools.ProteinTokenizer* method), 46
create_standardized_mol_id() (in module *pahelix.utils.compound_tools*), 43

D

data_list (in *pahelix.datasets.inmemory_dataset.InMemoryDataset* attribute), 25
DDiFeaturizer (class in *pahelix.featurizers.het_gnn_featurizer*), 32

F

forward() (in *pahelix.model_zoo.pretrain_gnns_model.AttrmaskModel* method), 33
forward() (in *pahelix.model_zoo.pretrain_gnns_model.PretrainGNNModel* method), 33
forward() (in *pahelix.model_zoo.pretrain_gnns_model.SupervisedModel* method), 33
forward() (in *pahelix.model_zoo.protein_sequence_model.ClassificationTaskModel* method), 34
forward() (in *pahelix.model_zoo.protein_sequence_model.LstmEncoderModel* method), 34
forward() (in *pahelix.model_zoo.protein_sequence_model.PretrainTaskModel* method), 34
forward() (in *pahelix.model_zoo.protein_sequence_model.ProteinEncoderModel* method), 35
forward() (in *pahelix.model_zoo.protein_sequence_model.ProteinModel* method), 35
forward() (in *pahelix.model_zoo.protein_sequence_model.RegressionTaskModel* method), 34
forward() (in *pahelix.model_zoo.protein_sequence_model.ResnetEncoderModel* method), 34

`forward()` (`pahelix.model_zoo.protein_sequence_model.SeqClassifierWithTaskModel` method), 34
`forward()` (`pahelix.model_zoo.protein_sequence_model.TransformerEncoderModel` method), 34
`forward()` (`pahelix.model_zoo.seq_vae_model.VAE` method), 35
`forward()` (`pahelix.networks.basic_block.Activation` method), 36
`forward()` (`pahelix.networks.basic_block.MLP` method), 36
`forward()` (`pahelix.networks.compound_encoder.AtomEmbedding` method), 37
`forward()` (`pahelix.networks.compound_encoder.AtomFloatEmbedding` method), 37
`forward()` (`pahelix.networks.compound_encoder.BondAngleFloatRBF` method), 37
`forward()` (`pahelix.networks.compound_encoder.BondEmbedding` method), 37
`forward()` (`pahelix.networks.compound_encoder.BondFloatRBF` method), 37
`forward()` (`pahelix.networks.gnn_block.GIN` method), 37
`forward()` (`pahelix.networks.gnn_block.GraphNorm` method), 38
`forward()` (`pahelix.networks.gnn_block.MeanPool` method), 38
`forward()` (`pahelix.networks.involution_block.Involution2D` method), 38
`forward_decoder()` (`pahelix.model_zoo.seq_vae_model.VAE` method), 35
`forward_encoder()` (`pahelix.model_zoo.seq_vae_model.VAE` method), 35

G

`gen_token_ids()` (`pahelix.utils.protein_tools.ProteinTokenizer` method), 46
`generate_scaffold()` (in module `pahelix.utils.splitters`), 47
`get_2d_atom_poses()` (`pahelix.utils.compound_tools.Compound3DKit` static method), 42
`get_atom_feature_dims()` (in module `pahelix.utils.compound_tools`), 43
`get_atom_feature_id()` (`pahelix.utils.compound_tools.CompoundKit` static method), 42
`get_atom_feature_size()` (`pahelix.utils.compound_tools.CompoundKit` static method), 42
`get_atom_names()` (`pahelix.utils.compound_tools.CompoundKit` static method), 42
`get_atom_poses()` (`pahelix.utils.compound_tools.Compound3DKit` static method), 42
`get_atom_value()` (`pahelix.utils.compound_tools.CompoundKit` static method), 42
`get_bond_feature_dims()` (in module `pahelix.utils.compound_tools`), 43
`get_bond_feature_id()` (`pahelix.utils.compound_tools.CompoundKit` static method), 42
`get_bond_feature_size()` (`pahelix.utils.compound_tools.CompoundKit` static method), 43
`get_bond_lengths()` (`pahelix.utils.compound_tools.Compound3DKit` static method), 42
`get_bond_value()` (`pahelix.utils.compound_tools.CompoundKit` static method), 43
`get_chembl_filtered_task_num()` (in module `pahelix.datasets.chembl_filtered_dataset`), 19
`get_data_loader()` (`pahelix.datasets.inmemory_dataset.InMemoryDataset` method), 25
`get_daylight_functional_group_counts()` (`pahelix.utils.compound_tools.CompoundKit` static method), 43
`get_default_bace_task_names()` (in module `pahelix.datasets.bace_dataset`), 17
`get_default_bbbp_task_names()` (in module `pahelix.datasets.bbbp_dataset`), 18
`get_default_clintox_task_names()` (in module `pahelix.datasets.clintox_dataset`), 20
`get_default_ddi_task_names()` (in module `pahelix.datasets.ddi_dataset`), 21
`get_default_dti_task_names()` (in module `pahelix.datasets.dti_dataset`), 22
`get_default_esol_task_names()` (in module `pahelix.datasets.esol_dataset`), 22
`get_default_freesolv_task_names()` (in module `pahelix.datasets.freesolv_dataset`), 23
`get_default_hiv_task_names()` (in module `pahelix.datasets.hiv_dataset`), 24
`get_default_lipophilicity_task_names()` (in module `pahelix.datasets.lipophilicity_dataset`), 26
`get_default_muv_task_names()` (in module `pahelix.datasets.muv_dataset`), 27
`get_default_ppi_task_names()` (in module `pahelix.datasets.ppi_dataset`), 28
`get_default_sider_task_names()` (in module `pahelix.datasets.sider_dataset`), 28

[get_default_tox21_task_names\(\)](#) (in module [pahelix.datasets.tox21_dataset](#)), 29
[get_default_toxcast_task_names\(\)](#) (in module [pahelix.datasets.toxcast_dataset](#)), 30
[get_esol_stat\(\)](#) (in module [pahelix.datasets.esol_dataset](#)), 22
[get_freesolv_stat\(\)](#) (in module [pahelix.datasets.freesolv_dataset](#)), 23
[get_gasteiger_partial_charges\(\)](#) (in module [pahelix.utils.compound_tools](#)), 43
[get_largest_mol\(\)](#) (in module [pahelix.utils.compound_tools](#)), 43
[get_lipophilicity_stat\(\)](#) (in module [pahelix.datasets.lipophilicity_dataset](#)), 26
[get_maccs_fingerprint\(\)](#) ([pahelix.utils.compound_tools.CompoundKit](#) static method), 43
[get_MMFF_atom_poses\(\)](#) ([pahelix.utils.compound_tools.Compound3DKit](#) static method), 42
[get_morgan2048_fingerprint\(\)](#) ([pahelix.utils.compound_tools.CompoundKit](#) static method), 43
[get_morgan_fingerprint\(\)](#) ([pahelix.utils.compound_tools.CompoundKit](#) static method), 43
[get_part_files\(\)](#) (in module [pahelix.utils.data_utils](#)), 45
[get_ring_size\(\)](#) ([pahelix.utils.compound_tools.CompoundKit](#) static method), 43
[get_superedge_angles\(\)](#) ([pahelix.utils.compound_tools.Compound3DKit](#) static method), 42
[GIN](#) (class in [pahelix.networks.gnn_block](#)), 37
[graph_dim](#) ([pahelix.model_zoo.pretrain_gnns_model.PretrainGNNModel](#) property), 33
[GraphNorm](#) (class in [pahelix.networks.gnn_block](#)), 38
I
[IndexSplitter](#) (class in [pahelix.utils.splitters](#)), 46
[init_weights\(\)](#) ([pahelix.model_zoo.protein_sequence_model.ResnetEncoderModel](#) method), 34
[init_weights\(\)](#) ([pahelix.model_zoo.protein_sequence_model.TransformerEncoderModel](#) method), 34
[InMemoryDataset](#) (class in [pahelix.datasets.inmemory_dataset](#)), 25
[Involution2D](#) (class in [pahelix.networks.involution_block](#)), 38
L
[load_bace_dataset\(\)](#) (in module [pahelix.datasets.bace_dataset](#)), 17
[load_bbbp_dataset\(\)](#) (in module [pahelix.datasets.bbbp_dataset](#)), 18
[load_chembl_filtered_dataset\(\)](#) (in module [pahelix.datasets.chembl_filtered_dataset](#)), 19
[load_clintox_dataset\(\)](#) (in module [pahelix.datasets.clintox_dataset](#)), 20
[load_davis_dataset\(\)](#) (in module [pahelix.datasets.davis_dataset](#)), 21
[load_ddi_dataset\(\)](#) (in module [pahelix.datasets.ddi_dataset](#)), 21
[load_dti_dataset\(\)](#) (in module [pahelix.datasets.dti_dataset](#)), 22
[load_esol_dataset\(\)](#) (in module [pahelix.datasets.esol_dataset](#)), 22
[load_freesolv_dataset\(\)](#) (in module [pahelix.datasets.freesolv_dataset](#)), 23
[load_hiv_dataset\(\)](#) (in module [pahelix.datasets.hiv_dataset](#)), 24
[load_json_config\(\)](#) (in module [pahelix.utils.basic_utils](#)), 42
[load_kiba_dataset\(\)](#) (in module [pahelix.datasets.kiba_dataset](#)), 26
[load_lipophilicity_dataset\(\)](#) (in module [pahelix.datasets.lipophilicity_dataset](#)), 26
[load_muv_dataset\(\)](#) (in module [pahelix.datasets.muv_dataset](#)), 27
[load_npz_to_data_list\(\)](#) (in module [pahelix.utils.data_utils](#)), 45
[load_ppi_dataset\(\)](#) (in module [pahelix.datasets.ppi_dataset](#)), 28
[load_sider_dataset\(\)](#) (in module [pahelix.datasets.sider_dataset](#)), 28
[load_tox21_dataset\(\)](#) (in module [pahelix.datasets.tox21_dataset](#)), 29
[load_toxcast_dataset\(\)](#) (in module [pahelix.datasets.toxcast_dataset](#)), 30
[load_zinc_dataset\(\)](#) (in module [pahelix.datasets.zinc_dataset](#)), 31
[lstm_encoder\(\)](#) (in module [pahelix.networks.lstm_block](#)), 39
[LstmEncoderModel](#) (class in [pahelix.model_zoo.protein_sequence_model](#)), 34
M
[MeanPool](#) (class in [pahelix.networks.gnn_block](#)), 38
[MLP](#) (class in [pahelix.networks.basic_block](#)), 36
module
[pahelix.datasets.bace_dataset](#), 17
[pahelix.datasets.bbbp_dataset](#), 18
[pahelix.datasets.chembl_filtered_dataset](#), 19
[pahelix.datasets.clintox_dataset](#), 20

pahelix.datasets.davis_dataset, 21
 pahelix.datasets.ddi_dataset, 21
 pahelix.datasets.dti_dataset, 22
 pahelix.datasets.esol_dataset, 22
 pahelix.datasets.freesolv_dataset, 23
 pahelix.datasets.hiv_dataset, 24
 pahelix.datasets.inmemory_dataset, 25
 pahelix.datasets.kiba_dataset, 26
 pahelix.datasets.lipophilicity_dataset, 26
 pahelix.datasets.muv_dataset, 27
 pahelix.datasets.ppi_dataset, 28
 pahelix.datasets.sider_dataset, 28
 pahelix.datasets.tox21_dataset, 29
 pahelix.datasets.toxcast_dataset, 30
 pahelix.datasets.zinc_dataset, 31
 pahelix.featurizers.het_gnn_featurizer, 32
 pahelix.featurizers.pretrain_gnn_featurizer, 32
 pahelix.model_zoo.pretrain_gnns_model, 33
 pahelix.model_zoo.protein_sequence_model, 34
 pahelix.networks.basic_block, 36
 pahelix.networks.compound_encoder, 37
 pahelix.networks.gnn_block, 37
 pahelix.networks.lstm_block, 39
 pahelix.networks.resnet_block, 40
 pahelix.networks.transformer_block, 40
 pahelix.utils.basic_utils, 42
 pahelix.utils.compound_tools, 42
 pahelix.utils.data_utils, 45
 pahelix.utils.language_model_tools, 45
 pahelix.utils.splitters, 46
 mol_to_geognn_graph_data() (in module *pahelix.utils.compound_tools*), 44
 mol_to_geognn_graph_data_MMFF3d() (in module *pahelix.utils.compound_tools*), 44
 mol_to_geognn_graph_data_raw3d() (in module *pahelix.utils.compound_tools*), 44
 mol_to_graph_data() (in module *pahelix.utils.compound_tools*), 44
 mp_pool_map() (in module *pahelix.utils.basic_utils*), 42
 multi_head_attention() (in module *pahelix.networks.transformer_block*), 40

N

new_mol_to_graph_data() (in module *pahelix.utils.compound_tools*), 44
 new_smiles_to_graph_data() (in module *pahelix.utils.compound_tools*), 44
 node_dim(*pahelix.model_zoo.pretrain_gnns_model.PretrainGNNSModel*, 33
 property), 33

num_nodes_stat() (in module *pahelix.featurizers.het_gnn_featurizer*), 32
 nx_graph_build() (in module *pahelix.featurizers.het_gnn_featurizer*), 32

P

pahelix.datasets.bace_dataset module, 17
 pahelix.datasets.bbbp_dataset module, 18
 pahelix.datasets.chembl_filtered_dataset module, 19
 pahelix.datasets.clintox_dataset module, 20
 pahelix.datasets.davis_dataset module, 21
 pahelix.datasets.ddi_dataset module, 21
 pahelix.datasets.dti_dataset module, 22
 pahelix.datasets.esol_dataset module, 22
 pahelix.datasets.freesolv_dataset module, 23
 pahelix.datasets.hiv_dataset module, 24
 pahelix.datasets.inmemory_dataset module, 25
 pahelix.datasets.kiba_dataset module, 26
 pahelix.datasets.lipophilicity_dataset module, 26
 pahelix.datasets.muv_dataset module, 27
 pahelix.datasets.ppi_dataset module, 28
 pahelix.datasets.sider_dataset module, 28
 pahelix.datasets.tox21_dataset module, 29
 pahelix.datasets.toxcast_dataset module, 30
 pahelix.datasets.zinc_dataset module, 31
 pahelix.featurizers.het_gnn_featurizer module, 32
 pahelix.featurizers.pretrain_gnn_featurizer module, 32
 pahelix.model_zoo.pretrain_gnns_model module, 33
 pahelix.model_zoo.protein_sequence_model module, 34
 pahelix.networks.basic_block module, 36

pahelix.networks.compound_encoder
 module, 37
 pahelix.networks.gnn_block
 module, 37
 pahelix.networks.lstm_block
 module, 39
 pahelix.networks.resnet_block
 module, 40
 pahelix.networks.transformer_block
 module, 40
 pahelix.utils.basic_utils
 module, 42
 pahelix.utils.compound_tools
 module, 42
 pahelix.utils.data_utils
 module, 45
 pahelix.utils.language_model_tools
 module, 45
 pahelix.utils.splitters
 module, 46
 positionwise_feed_forward() (in module *pahelix.networks.transformer_block*), 40
 pre_post_process_layer() (in module *pahelix.networks.pre_post_process*), 39
 PretrainGNNModel (class in *pahelix.model_zoo.pretrain_gnns_model*), 33
 PretrainTaskModel (class in *pahelix.model_zoo.protein_sequence_model*), 34
 ProteinCriterion (class in *pahelix.model_zoo.protein_sequence_model*), 35
 ProteinEncoderModel (class in *pahelix.model_zoo.protein_sequence_model*), 34
 ProteinModel (class in *pahelix.model_zoo.protein_sequence_model*), 35
 ProteinTokenizer (class in *pahelix.utils.protein_tools*), 45

R

RandomScaffoldSplitter (class in *pahelix.utils.splitters*), 47
 RandomSplitter (class in *pahelix.utils.splitters*), 46
 rdchem_enum_to_list() (in module *pahelix.utils.compound_tools*), 44
 RegressionTaskModel (class in *pahelix.model_zoo.protein_sequence_model*), 34
 resnet_encoder() (in module *pahelix.networks.resnet_block*), 40
 ResnetEncoderModel (class in *pahelix.model_zoo.protein_sequence_model*),

34

S

safe_index() (in module *pahelix.utils.compound_tools*), 44
 sample() (*pahelix.model_zoo.seq_vae_model.VAE* method), 35
 sample_z_prior() (*pahelix.model_zoo.seq_vae_model.VAE* method), 35
 save_data() (*pahelix.datasets.inmemory_dataset.InMemoryDataset* method), 26
 save_data_list_to_npz() (in module *pahelix.utils.data_utils*), 45
 ScaffoldSplitter (class in *pahelix.utils.splitters*), 47
 SeqClassificationTaskModel (class in *pahelix.model_zoo.protein_sequence_model*), 34
 split() (*pahelix.utils.splitters.IndexSplitter* method), 47
 split() (*pahelix.utils.splitters.RandomScaffoldSplitter* method), 47
 split() (*pahelix.utils.splitters.RandomSplitter* method), 46
 split() (*pahelix.utils.splitters.ScaffoldSplitter* method), 47
 split_rdkit_mol_obj() (in module *pahelix.utils.compound_tools*), 44
 SupervisedCollateFn (class in *pahelix.featurizers.pretrain_gnn_featurizer*), 32
 SupervisedModel (class in *pahelix.model_zoo.pretrain_gnns_model*), 33
 SupervisedTransformFn (class in *pahelix.featurizers.pretrain_gnn_featurizer*), 32

T

tensor2string() (*pahelix.model_zoo.seq_vae_model.VAE* method), 35
 tokenize() (*pahelix.utils.protein_tools.ProteinTokenizer* method), 46
 transform() (*pahelix.datasets.inmemory_dataset.InMemoryDataset* method), 26
 transformer_encoder() (in module *pahelix.networks.transformer_block*), 40
 transformer_encoder_layer() (in module *pahelix.networks.transformer_block*), 41
 TransformerEncoderModel (class in *pahelix.model_zoo.protein_sequence_model*), 34

V

VAE (class in *pahelix.model_zoo.seq_vae_model*), 35