torch_geometric.data

class **Data**(*x=None, edge_index=None, edge_attr=None, y=None, pos=None, norm=None, face=None, **kwargs*) [source]

A plain old python object modeling a single graph with various (optional) attributes:

Parameters:

- **x** (*Tensor, optional*) – Node feature matrix with shape `[num_nodes, num_node_features]`. (default: `None`)
- **edge_index** (*LongTensor, optional*) – Graph connectivity in COO format with shape `[2, num_edges]`. (default: `None`)
- **edge_attr** (*Tensor, optional*) – Edge feature matrix with shape `[num_edges, num_edge_features]`. (default: `None`)
- **y** (*Tensor, optional*) – Graph or node targets with arbitrary shape. (default: `None`)
- **pos** (*Tensor, optional*) – Node position matrix with shape `[num_nodes, num_dimensions]`. (default: `None`)
- **norm** (*Tensor, optional*) – Normal vector matrix with shape `[num_nodes, num_dimensions]`. (default: `None`)
- **face** (*LongTensor, optional*) – Face adjacency matrix with shape `[3, num_faces]`. (default: `None`)

The data object is not restricted to these attributes and can be extented by any other additional data.

Example:

```
data = Data(x=x, edge_index=edge_index)
data.train_idx = torch.tensor([...], dtype=torch.long)
data.test_mask = torch.tensor([...], dtype=torch.uint8)
```

__call__(**keys*) [source]

Iterates over all attributes `*keys` in the data, yielding their attribute names and content. If `*keys` is not given this method will iterative over all present attributes.

__cat_dim__(*key, value*) [source]

Returns the dimension for which `value` of attribute `key` will get concatenated when creating batches.

Note

This method is for internal use only, and should only be overridden if the batch concatenation process is corrupted for a specific data attribute.

`__contains__`(*key*) [source]

Returns `True`, if the attribute `key` is present in the data.

`__cumsum__`(*key*, *value*) [source]

If `True`, `value` of attribute `key` is cumulatively summed up when creating batches.

> **Note**
>
> This method is for internal use only, and should only be overridden if the batch concatenation process is corrupted for a specific data attribute.

`__getitem__`(*key*) [source]

Gets the data of the attribute `key`.

`__iter__`() [source]

Iterates over all present attributes in the data, yielding their attribute names and content.

`__len__`() [source]

Returns the number of all present attributes.

`__setitem__`(*key*, *value*) [source]

Sets the attribute `key` to `value`.

`apply`(*func*, **keys*) [source]

Applies the function `func` to all tensor attributes `*keys`. If `*keys` is not given, `func` is applied to all present attributes.

`clone`() [source]

`contains_isolated_nodes`() [source]

Returns `True`, if the graph does not contain isolated nodes.

`contains_self_loops`() [source]

Returns `True`, if the graph does not contain self-loops.

contiguous(**keys*) [source]

Ensures a contiguous memory layout for all attributes `*keys`. If `*keys` is not given, all present attributes are ensured to have a contiguous memory layout.

static **from_dict**(*dictionary*) [source]

Creates a data object from a python dictionary.

is_coalesced() [source]

Returns `True`, if edge indices are ordered and do not contain duplicate entries.

is_directed() [source]

Returns `True`, if graph edges are directed.

is_undirected() [source]

Returns `True`, if graph edges are undirected.

keys

Returns all names of graph attributes.

num_edges

Returns the number of edges in the graph.

num_faces

Returns the number of faces in the mesh.

num_features

Returns the number of features per node in the graph.

num_nodes

Returns or sets the number of nodes in the graph.

Note

The number of nodes in your data object is typically automatically inferred, *e.g.*, when node features `x` are present. In some cases however, a graph may only be given by its edge indices `edge_index`. PyTorch Geometric then *guesses* the number of nodes according to `edge_index.max().item() + 1`, but in case there exists isolated nodes, this number has not to be correct and can therefore result in unexpected

batch-wise behavior. Thus, we recommend to set the number of nodes in your data object explicitly via `data.num_nodes = ...`. You will be given a warning that requests you to do so.

to(*device*, **keys*) [source]

Performs tensor dtype and/or device conversion to all attributes `*keys`. If `*keys` is not given, the conversion is applied to all present attributes.

class **Batch**(*batch=None*, ***kwargs*) [source]

A plain old python object modeling a batch of graphs as one big (dicconnected) graph. With `torch_geometric.data.Data` being the base class, all its methods can also be used here. In addition, single graphs can be reconstructed via the assignment vector `batch`, which maps each node to its respective graph identifier.

static **from_data_list**(*data_list*, *follow_batch=[]*) [source]

Constructs a batch object from a python list holding `torch_geometric.data.Data` objects. The assignment vector `batch` is created on the fly. Additionally, creates assignment batch vectors for each key in `follow_batch`.

num_graphs

Returns the number of graphs in the batch.

class **Dataset**(*root*, *transform=None*, *pre_transform=None*, *pre_filter=None*) [source]

Dataset base class for creating graph datasets. See here for the accompanying tutorial.

Parameters:

- **root** (*string*) – Root directory where the dataset should be saved.
- **transform** (*callable*, *optional*) – A function/transform that takes in an `torch_geometric.data.Data` object and returns a transformed version. The data object will be transformed before every access. (default: `None`)
- **pre_transform** (*callable*, *optional*) – A function/transform that takes in an `torch_geometric.data.Data` object and returns a transformed version. The data object will be transformed before being saved to disk. (default: `None`)
- **pre_filter** (*callable*, *optional*) – A function that takes in an `torch_geometric.data.Data` object and returns a boolean value, indicating whether the data object should be included in the final dataset. (default: `None`)

__getitem__(*idx*) [source]

Gets the data object at index `idx` and transforms it (in case a `self.transform` is given).

__len__() [source]

The number of examples in the dataset.

download() [source]

Downloads the dataset to the `self.raw_dir` folder.

get(*idx*) [source]

Gets the data object at index `idx`.

num_features

Returns the number of features per node in the graph.

process() [source]

Processes the dataset to the `self.processed_dir` folder.

processed_file_names

The name of the files to find in the `self.processed_dir` folder in order to skip the processing.

processed_paths

The filepaths to find in the `self.processed_dir` folder in order to skip the processing.

raw_file_names

The name of the files to find in the `self.raw_dir` folder in order to skip the download.

raw_paths

The filepaths to find in order to skip the download.

---

class **InMemoryDataset**(*root, transform=None, pre_transform=None, pre_filter=None*) [source]

Dataset base class for creating graph datasets which fit completely into memory. See here for the accompanying tutorial.

Parameters:

- **root** (*string*) – Root directory where the dataset should be saved.
- **transform** (*callable, optional*) – A function/transform that takes in an `torch_geometric.data.Data` object and returns a transformed version. The data object will be transformed before every access. (default: `None`)
- **pre_transform** (*callable, optional*) – A function/transform that takes in an `torch_geometric.data.Data` object and returns a transformed version. The data object will be transformed before being saved to disk. (default: `None`)
- **pre_filter** (*callable, optional*) – A function that takes in an `torch_geometric.data.Data` object and returns a boolean value, indicating whether the data object should be included in the final dataset. (default: `None`)

`__getitem__`(*idx*) [source]

Gets the data object at index `idx` and transforms it (in case a `self.transform` is given). Returns a data object, if `idx` is a scalar, and a new dataset in case `idx` is a slicing object, *e.g.*, `[2:5]`, a LongTensor or a ByteTensor.

`__indexing__`(*index*) [source]

`__len__`() [source]

The number of examples in the dataset.

`collate`(*data_list*) [source]

Collates a python list of data objects to the internal storage format of `torch_geometric.data.InMemoryDataset`.

`download`() [source]

Downloads the dataset to the `self.raw_dir` folder.

`get`(*idx*) [source]

Gets the data object at index `idx`.

`num_classes`

The number of classes in the dataset.

`process`() [source]

Processes the dataset to the `self.processed_dir` folder.

`processed_file_names`

The name of the files to find in the `self.processed_dir` folder in order to skip the processing.

raw_file_names

The name of the files to find in the `self.raw_dir` folder in order to skip the download.

shuffle() [source]

Randomly shuffles the examples in the dataset.

---

class **DataLoader**(*dataset, batch_size=1, shuffle=True, follow_batch=[], **kwargs*) [source]

Data loader which merges data objects from a `torch_geometric.data.dataset` to a mini-batch.

Parameters:

- **dataset** (*Dataset*) – The dataset from which to load the data.
- **batch_size** (*int*, *optional*) – How may samples per batch to load. (default: `1`)
- **shuffle** (*bool*, *optional*) – If set to `True`, the data will be reshuffled at every epoch (default: `True`)
- **follow_batch** (*list or tuple*, *optional*) – Creates assignment batch vectors for each key in the list. (default: `[]`)

---

class **DataListLoader**(*dataset, batch_size=1, shuffle=True, **kwargs*) [source]

Data loader which merges data objects from a `torch_geometric.data.dataset` to a python list.

> **Note**
>
> This data loader should be used for multi-gpu support via `torch_geometric.nn.DataParallel`.

Parameters:

- **dataset** (*Dataset*) – The dataset from which to load the data.
- **batch_size** (*int*, *optional*) – How may samples per batch to load. (default: `1`)
- **shuffle** (*bool*, *optional*) – If set to `True`, the data will be reshuffled at every epoch (default: `True`)

---

class **DenseDataLoader**(*dataset, batch_size=1, shuffle=True, **kwargs*) [source]

Data loader which merges data objects from a `torch_geometric.data.dataset` to a mini-batch.

Note

To make use of this data loader, all graphs in the dataset needs to have the same shape for each its attributes. Therefore, this data loader should only be used when working with *dense* adjacency matrices.

Parameters:

- **dataset** (*Dataset*) – The dataset from which to load the data.
- **batch_size** (*int*, *optional*) – How may samples per batch to load. (default: `1`)
- **shuffle** (*bool*, *optional*) – If set to `True`, the data will be reshuffled at every epoch (default: `True`)

`download_url`(*url, folder, log=True*) [source]

Downloads the content of an URL to a specific folder.

Parameters:

- **url** (*string*) – The url.
- **folder** (*string*) – The folder.
- **log** (*bool*, *optional*) – If `False`, will not print anything to the console. (default: `True`)

`extract_tar`(*path, folder, mode='r:gz', log=True*) [source]

Extracts a tar archive to a specific folder.

Parameters:

- **path** (*string*) – The path to the tar archive.
- **folder** (*string*) – The folder.
- **mode** (*string*, *optional*) – The compression mode. (default: `"r:gz"`)
- **log** (*bool*, *optional*) – If `False`, will not print anything to the console. (default: `True`)

`extract_zip`(*path, folder, log=True*) [source]

Extracts a zip archive to a specific folder.

Parameters:

- **path** (*string*) – The path to the tar archive.
- **folder** (*string*) – The folder.
- **log** (*bool*, *optional*) – If `False`, will not print anything to the console. (default: `True`)

`extract_bz2`(*path, folder, log=True*) [source]

---

extract_gz(*path*, *folder*, *log=True*) [source]