# CONTENTS

1 Contents .................................................. 3
  1.1 API ...................................................... 3

Python Module Index .......................................... 25

Index ........................................................... 27
CGLB is a benchmark for continual graph learning (CGL). We also provide a comprehensive toolkit for benchmarking different CGL methods, and the instructions are included in this documentation.

Our GitHub repository CGLB
1.1 API

1.1.1 NCGL Baselines

Bare model

```python
class NCGL.Baselines.bare_model.NET(*args: Any, **kwargs: Any)
    Bare model baseline for NCGL tasks
    Parameters
    • model – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
    • task_manager – Mainly serves to store the indices of the output dimensions corresponding to each task
    • args – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.
```

```python
observe(args, g, features, labels, t, train_ids, ids_per_cls, dataset)
    The method for learning the given tasks under the class-IL setting.
    Parameters
    • args – Same as the args in __init__().
    • g – The graph of the current task.
    • features – Node features of the current task.
    • labels – Labels of the nodes in the current task.
    • t – Index of the current task.
    • train_ids – The indices of the nodes participating in the training.
    • ids_per_cls – Indices of the nodes in each class (not in use in the current baseline).
    • dataset – The entire dataset (not in use in the current baseline).
```

```python
observe_class_IL_batch(args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset)
    The method for learning the given tasks under the class-IL setting with mini-batch training.
    Parameters
```
• **args** – Same as the args in `__init__()`.
• **g** – The graph of the current task.
• **dataloader** – The data loader for mini-batch training.
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (currently not in use).
• **dataset** – The entire dataset (currently not in use).

**observe_task_IL** *(args, g, features, labels, t, train_ids, ids_per_cls, dataset)*

The method for learning the given tasks under the task-IL setting.

**Parameters**

• **args** – Same as the args in `__init__()`.
• **g** – The graph of the current task.
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
• **dataset** – The entire dataset (not in use in the current baseline).

**observe_task_IL_batch** *(args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset)*

The method for learning the given tasks under the task-IL setting with mini-batch training.

**Parameters**

• **args** – Same as the args in `__init__()`.
• **g** – The graph of the current task.
• **dataloader** – The data loader for mini-batch training
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (currently not in use).
• **dataset** – The entire dataset (currently not in use).
ER-GNN

class NCGL.Baselines.ergnn_model.NET(*args: Any, **kwargs: Any)

ER-GNN baseline for NCGL tasks

Parameters

• **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
• **task_manager** – Mainly serves to store the indices of the output dimensions corresponding to each task
• **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

observe(args, g, features, labels, t, train_ids, ids_per_cls, dataset)

The method for learning the given tasks under the class-IL setting.

Parameters

• **args** – Same as the args in __init__().
• **g** – The graph of the current task.
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class.
• **dataset** – The entire dataset.

observe_class_IL_batch(args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset)

The method for learning the given tasks under the class-IL setting with mini-batch training.

Parameters

• **args** – Same as the args in __init__().
• **g** – The graph of the current task.
• **dataloader** – The data loader for mini-batch training
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (currently not in use).
• **dataset** – The entire dataset (currently not in use).

observe_task_IL(args, g, features, labels, t, train_ids, ids_per_cls, dataset)

The method for learning the given tasks under the task-IL setting.

Parameters

• **args** – Same as the args in __init__().
• `g` – The graph of the current task.
• `features` – Node features of the current task.
• `labels` – Labels of the nodes in the current task.
• `t` – Index of the current task.
• `train_ids` – The indices of the nodes participating in the training.
• `ids_per_cls` – Indices of the nodes in each class.
• `dataset` – The entire dataset.

`observe_task_IL_batch(args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset)`

The method for learning the given tasks under the task-IL setting with mini-batch training.

Parameters
• `args` – Same as the args in `__init__()`.
• `g` – The graph of the current task.
• `dataloader` – The data loader for mini-batch training
• `features` – Node features of the current task.
• `labels` – Labels of the nodes in the current task.
• `t` – Index of the current task.
• `train_ids` – The indices of the nodes participating in the training.
• `ids_per_cls` – Indices of the nodes in each class (currently not in use).
• `dataset` – The entire dataset (currently not in use).

EWC

class NCGL.Baselines.ewc_model.NET(*args: Any, **kwargs: Any)

EWC baseline for NCGL tasks

Parameters
• `model` – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
• `task_manager` – Mainly serves to store the indices of the output dimensions corresponding to each task
• `args` – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

`observe(args, g, features, labels, t, train_ids, ids_per_cls, dataset)`

The method for learning the given tasks under the class-IL setting.

Parameters
• `args` – Same as the args in `__init__()`.
• `g` – The graph of the current task.
• `features` – Node features of the current task.
• `labels` – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
• **dataset** – The entire dataset (not in use in the current baseline).

**observe_class_IL_batch**(*args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset*)

The method for learning the given tasks under the class-IL setting with mini-batch training.

**Parameters**

• **args** – Same as the args in **__init__**().
• **g** – The graph of the current task.
• **dataloader** – The data loader for mini-batch training
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (currently not in use).
• **dataset** – The entire dataset (currently not in use).

**observe_task_IL**(*args, g, features, labels, t, train_ids, ids_per_cls, dataset*)

The method for learning the given tasks under the task-IL setting.

**Parameters**

• **args** – Same as the args in **__init__**().
• **g** – The graph of the current task.
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
• **dataset** – The entire dataset (not in use in the current baseline).

**observe_task_IL_batch**(*args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset*)

The method for learning the given tasks under the task-IL setting with mini-batch training.

**Parameters**

• **args** – Same as the args in **__init__**().
• **g** – The graph of the current task.
• **dataloader** – The data loader for mini-batch training
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (currently not in use).
- **dataset** – The entire dataset (currently not in use).

**GEM**

class NCGL.Baselines.gem_model.NET(*args: Any, **kwargs: Any)

GEM baseline for NCGL tasks

**Parameters**
- **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
- **task_manager** – Mainly serves to store the indices of the output dimensions corresponding to each task
- **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

**observe**(*args, g, features, labels, t, train_ids, ids_per_cls, dataset*)

The method for learning the given tasks under the class-IL setting.

**Parameters**
- **args** – Same as the args in __init__().
- **g** – The graph of the current task.
- **features** – Node features of the current task.
- **labels** – Labels of the nodes in the current task.
- **t** – Index of the current task.
- **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
- **dataset** – The entire dataset (not in use in the current baseline).

**observe_class_IL_batch**(*args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset*)

The method for learning the given tasks under the class-IL setting with mini-batch training.

**Parameters**
- **args** – Same as the args in __init__().
- **g** – The graph of the current task.
- **dataloader** – The data loader for mini-batch training
- **features** – Node features of the current task.
- **labels** – Labels of the nodes in the current task.
- **t** – Index of the current task.
- **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (currently not in use).
- **dataset** – The entire dataset (currently not in use).
observe_task_IL(args, g, features, labels, t, train_ids, ids_per_cls, dataset)

The method for learning the given tasks under the task-IL setting.

**Parameters**
- **args** – Same as the args in __init__().
- **g** – The graph of the current task.
- **features** – Node features of the current task.
- **labels** – Labels of the nodes in the current task.
- **t** – Index of the current task.
- **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
- **dataset** – The entire dataset (not in use in the current baseline).

observe_task_IL_batch(args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset)

The method for learning the given tasks under the task-IL setting with mini-batch training.

**Parameters**
- **args** – Same as the args in __init__().
- **g** – The graph of the current task.
- **dataloader** – The data loader for mini-batch training
- **features** – Node features of the current task.
- **labels** – Labels of the nodes in the current task.
- **t** – Index of the current task.
- **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (currently not in use).
- **dataset** – The entire dataset (currently not in use).

Joint

class NCGL.Baselines.joint_replay_all.NET(*args: Any, **kwargs: Any)

Jointly trained model baseline for NCGL tasks

In this baseline, when ever a new task comes, the model will be jointly trained on all existing tasks.

**Parameters**
- **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
- **task_manager** – Mainly serves to store the indices of the output dimensions corresponding to each task
- **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.
**observe**(*args, gs, featuers, labelss, t, train_idss, ids_per_clss, dataset*)

The method for learning the given tasks under the class-IL setting.

**Parameters**
- **args** – Same as the args in __init__().
- **gs** – The graphs of all the existing tasks to be jointly trained on.
- **featuress** – Node features of the nodes in all the existing tasks.
- **labelss** – Labels of the nodes in all the existing tasks.
- **t** – Index of the newest task.
- **train_idss** – The indices of the nodes participating in the training in all the existing tasks.
- **ids_per_clss** – Indices of the nodes in each class of all the existing tasks.
- **dataset** – The entire dataset (not in use in the current baseline).

**observe_class_IL_batch**(*args, gs, dataloader, featuers, labelss, t, train_idss, ids_per_clss, dataset*)

The method for learning the given tasks under the class-IL setting with mini-batch training.

**Parameters**
- **args** – Same as the args in __init__().
- **gs** – The graphs of all the existing tasks to be jointly trained on.
- **dataloader** – The data loader for mini-batch training.
- **featuress** – Node features of the nodes in all the existing tasks.
- **labelss** – Labels of the nodes in all the existing tasks.
- **t** – Index of the newest task.
- **train_idss** – The indices of the nodes participating in the training in all the existing tasks.
- **ids_per_clss** – Indices of the nodes in each class of all the existing tasks.
- **dataset** – The entire dataset.

**observe_class_IL_crsedge**(*args, g, features, labels_all, t, train_ids, ids_per_cls_all, dataset*)

The method for learning the given tasks under the class-IL setting with inter-task edges.

**Parameters**
- **args** – Same as the args in __init__().
- **g** – The graphs of all the existing tasks to be jointly trained on.
- **features** – Node features of the nodes in all the existing tasks.
- **labels_all** – Labels of the nodes in all the existing tasks.
- **t** – Index of the newest task.
- **train_ids** – The indices of the nodes participating in the training in all the existing tasks.
- **ids_per_cls_all** – Indices of the nodes in each class of all the existing tasks.
- **dataset** – The entire dataset (not in use in the current baseline).
**observe_task_IL**\( (\text{args, gs, features, labelss, t, train_idss, ids_per_clss, dataset}) \)

The method for learning the given tasks under the task-IL setting.

**Parameters**

- **args** – Same as the args in **__init__()**.
- **gs** – The graphs of all the existing tasks to be jointly trained on.
- **features** – Node features of the nodes in all the existing tasks.
- **labelss** – Labels of the nodes in all the existing tasks.
- **t** – Index of the newest task.
- **train_idss** – The indices of the nodes participating in the training in all the existing tasks.
- **ids_per_clss** – Indices of the nodes in each class of all the existing tasks.
- **dataset** – The entire dataset (not in use in the current baseline).

**observe_task_IL_batch**\( (\text{args, gs, dataloader, features, labelss, t, train_idss, ids_per_clss, dataset}) \)

The method for learning the given tasks under the task-IL setting with mini-batch training.

**Parameters**

- **args** – Same as the args in **__init__()**.
- **gs** – The graphs of all the existing tasks to be jointly trained on.
- **dataloader** – The data loader for mini-batch training.
- **features** – Node features of the nodes in all the existing tasks.
- **labelss** – Labels of the nodes in all the existing tasks.
- **t** – Index of the newest task.
- **train_idss** – The indices of the nodes participating in the training in all the existing tasks.
- **ids_per_clss** – Indices of the nodes in each class of all the existing tasks.
- **dataset** – The entire dataset (not in use in the current baseline).

**observe_task_IL_crsedge**\( (\text{args, g, features, labels_all, t, train_ids, ids_per_cls_all, dataset}) \)

The method for learning the given tasks under the task-IL setting with inter-task edges.

**Parameters**

- **args** – Same as the args in **__init__()**.
- **g** – The graphs of all the existing tasks to be jointly trained on.
- **features** – Node features of the nodes in all the existing tasks.
- **labels_all** – Labels of the nodes in all the existing tasks.
- **t** – Index of the newest task.
- **train_ids** – The indices of the nodes participating in the training in all the existing tasks.
- **ids_per_cls_all** – Indices of the nodes in each class of all the existing tasks.
- **dataset** – The entire dataset (not in use in the current baseline).
LwF

class NCGL.Baselines.lwf_model.NET(*args: Any, **kwargs: Any)
    LwF baseline for NCGL tasks

    Parameters
    • **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
    • **task_manager** – Mainly serves to store the indices of the output dimensions corresponding to each task
    • **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

    observe(args, g, features, labels, t, prev_model, train_ids, ids_per_cls, dataset)
    The method for learning the given tasks under the class-IL setting.

    Parameters
    • **args** – Same as the args in __init__().
    • **g** – The graph of the current task.
    • **features** – Node features of the current task.
    • **labels** – Labels of the nodes in the current task.
    • **t** – Index of the current task.
    • **prev_model** – The model obtained after learning the previous task.
    • **train_ids** – The indices of the nodes participating in the training.
    • **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
    • **dataset** – The entire dataset (not in use in the current baseline).

    observe_class_IL_batch(args, g, dataloader, features, labels, t, prev_model, train_ids, ids_per_cls, dataset)
    The method for learning the given tasks under the class-IL setting with mini-batch training.

    Parameters
    • **args** – Same as the args in __init__().
    • **g** – The graph of the current task.
    • **dataloader** – The data loader for mini-batch training
    • **features** – Node features of the current task.
    • **labels** – Labels of the nodes in the current task.
    • **t** – Index of the current task.
    • **prev_model** – The model obtained after learning the previous task.
    • **train_ids** – The indices of the nodes participating in the training.
    • **ids_per_cls** – Indices of the nodes in each class (currently not in use).
    • **dataset** – The entire dataset (currently not in use).
observe_task_IL(args, g, features, labels, t, prev_model, train_ids, ids_per_cls, dataset)

The method for learning the given tasks under the task-IL setting.

Parameters
- **args** – Same as the args in __init__().
- **g** – The graph of the current task.
- **features** – Node features of the current task.
- **labels** – Labels of the nodes in the current task.
- **t** – Index of the current task.
- **prev_model** – The model obtained after learning the previous task.
- **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
- **dataset** – The entire dataset (not in use in the current baseline).

observe_task_IL_batch(args, g, dataloader, features, labels, t, prev_model, train_ids, ids_per_cls, dataset)

The method for learning the given tasks under the task-IL setting with mini-batch training.

Parameters
- **args** – Same as the args in __init__().
- **g** – The graph of the current task.
- **dataloader** – The data loader for mini-batch training
- **features** – Node features of the current task.
- **labels** – Labels of the nodes in the current task.
- **t** – Index of the current task.
- **prev_model** – The model obtained after learning the previous task.
- **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (currently not in use).
- **dataset** – The entire dataset (currently not in use).

MAS

class NCGL.Baselines.mas_model.NET(*args: Any, **kwargs: Any)

MAS baseline for NCGL tasks

Parameters
- **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
- **task_manager** – Mainly serves to store the indices of the output dimensions corresponding to each task
- **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.
observe\((\text{args}, \text{g}, \text{features}, \text{labels}, \text{t}, \text{train\_ids}, \text{ids\_per\_cls}, \text{dataset})\)

The method for learning the given tasks under the class-IL setting.

Parameters
- \textbf{args} – Same as the args in \texttt{__init__}().
- \textbf{g} – The graph of the current task.
- \textbf{features} – Node features of the current task.
- \textbf{labels} – Labels of the nodes in the current task.
- \textbf{t} – Index of the current task.
- \textbf{train\_ids} – The indices of the nodes participating in the training.
- \textbf{ids\_per\_cls} – Indices of the nodes in each class (not in use in the current baseline).
- \textbf{dataset} – The entire dataset (not in use in the current baseline).

observe\_class\_IL\_batch\((\text{args}, \text{g}, \text{dataloader}, \text{features}, \text{labels}, \text{t}, \text{train\_ids}, \text{ids\_per\_cls}, \text{dataset})\)

The method for learning the given tasks under the class-IL setting with mini-batch training.

Parameters
- \textbf{args} – Same as the args in \texttt{__init__}().
- \textbf{g} – The graph of the current task.
- \textbf{dataloader} – The data loader for mini-batch training
- \textbf{features} – Node features of the current task.
- \textbf{labels} – Labels of the nodes in the current task.
- \textbf{t} – Index of the current task.
- \textbf{train\_ids} – The indices of the nodes participating in the training.
- \textbf{ids\_per\_cls} – Indices of the nodes in each class (currently not in use).
- \textbf{dataset} – The entire dataset (currently not in use).

observe\_task\_IL\((\text{args}, \text{g}, \text{features}, \text{labels}, \text{t}, \text{train\_ids}, \text{ids\_per\_cls}, \text{dataset})\)

The method for learning the given tasks under the task-IL setting.

Parameters
- \textbf{args} – Same as the args in \texttt{__init__}().
- \textbf{g} – The graph of the current task.
- \textbf{features} – Node features of the current task.
- \textbf{labels} – Labels of the nodes in the current task.
- \textbf{t} – Index of the current task.
- \textbf{train\_ids} – The indices of the nodes participating in the training.
- \textbf{ids\_per\_cls} – Indices of the nodes in each class (not in use in the current baseline).
- \textbf{dataset} – The entire dataset (not in use in the current baseline).
**observe_task_IL_batch**(*args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset*)

The method for learning the given tasks under the task-IL setting with mini-batch training.

**Parameters**

- **args** – Same as the args in **init**().
- **g** – The graph of the current task.
- **dataloader** – The data loader for mini-batch training
- **features** – Node features of the current task.
- **labels** – Labels of the nodes in the current task.
- **t** – Index of the current task.
- **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (currently not in use).
- **dataset** – The entire dataset (currently not in use).

**TWP**

class NCGL.Baselines.twp_model.NET(*args: Any, **kwargs: Any*)

Bare model baseline for NCGL tasks

**Parameters**

- **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
- **task_manager** – Mainly serves to store the indices of the output dimensions corresponding to each task
- **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

**observe**(*args, g, features, labels, t, train_ids, ids_per_cls, dataset*)

The method for learning the given tasks under the class-IL setting.

**Parameters**

- **args** – Same as the args in **init**().
- **g** – The graph of the current task.
- **features** – Node features of the current task.
- **labels** – Labels of the nodes in the current task.
- **t** – Index of the current task.
- **train_ids** – The indices of the nodes participating in the training.
- **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
- **dataset** – The entire dataset (not in use in the current baseline).

**observe_class_IL_batch**(*args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset*)

The method for learning the given tasks under the class-IL setting with mini-batch training.

**Parameters**
• **args** – Same as the args in `__init__()`.
• **g** – The graph of the current task.
• **dataloader** – The data loader for mini-batch training
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (currently not in use).
• **dataset** – The entire dataset (currently not in use).

**observe_task_IL** *(args, g, features, labels, t, train_ids, ids_per_cls, dataset)*

The method for learning the given tasks under the task-IL setting.

**Parameters**
• **args** – Same as the args in `__init__()`.
• **g** – The graph of the current task.
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
• **dataset** – The entire dataset (not in use in the current baseline).

**observe_task_IL_batch** *(args, g, dataloader, features, labels, t, train_ids, ids_per_cls, dataset)*

The method for learning the given tasks under the task-IL setting with mini-batch training.

**Parameters**
• **args** – Same as the args in `__init__()`.
• **g** – The graph of the current task.
• **dataloader** – The data loader for mini-batch training
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (currently not in use).
• **dataset** – The entire dataset (currently not in use).
1.1.2 GCGL Baselines

Bare model

class GCGL.Baselines.bare_model.NET(*args: Any, **kwargs: Any)

Bare model baseline for GCGL tasks

Parameters

- **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
- **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

observe(data_loader, loss_criterion, task_i, args)

The method for learning the given tasks under the task-IL setting with multi-label classification datasets.

Parameters

- **data_loader** – The data loader for mini-batch training.
- **loss_criterion** – The loss function.
- **task_i** – Index of the current task.
- **args** – Same as the args in __init__().

observe_clsIL(data_loader, loss_criterion, task_i, args)

The method for learning the given tasks under the class-IL setting with multi-class classification datasets.

Parameters

- **data_loader** – The data loader for mini-batch training.
- **loss_criterion** – The loss function.
- **task_i** – Index of the current task.
- **args** – Same as the args in __init__().

observe_tskIL_multicls(data_loader, loss_criterion, task_i, args)

The method for learning the given tasks under the task-IL setting with multi-class classification datasets.

Parameters

- **data_loader** – The data loader for mini-batch training.
- **loss_criterion** – The loss function.
- **task_i** – Index of the current task.
- **args** – Same as the args in __init__().
EWC

class GCGL.Baselines.ewc_model.NET(*args: Any, **kwargs: Any)

EWC baseline for GCGL tasks

Parameters

• **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
• **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

observe(*data_loader, loss_criterion, task_i, args*)

The method for learning the given tasks under the task-IL setting with multi-label classification datasets.

Parameters

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

observe_clsIL(*data_loader, loss_criterion, task_i, args*)

The method for learning the given tasks under the class-IL setting with multi-class classification datasets.

Parameters

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

observe_tskIL_multiclass(*data_loader, loss_criterion, task_i, args*)

The method for learning the given tasks under the task-IL setting with multi-class classification datasets.

Parameters

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

GEM

class GCGL.Baselines.gem_model.NET(*args: Any, **kwargs: Any)

Bare model baseline for GCGL tasks

Parameters

• **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
• **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

**observe**(data_loader, loss_criterion, task_i, args)

The method for learning the given tasks under the task-IL setting with multi-label classification datasets.

**Parameters**

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**observe_clsIL**(data_loader, loss_criterion, task_i, args)

The method for learning the given tasks under the class-IL setting with multi-class classification datasets.

**Parameters**

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**observe_tskIL_multicls**(data_loader, loss_criterion, task_i, args)

The method for learning the given tasks under the task-IL setting with multi-class classification datasets.

**Parameters**

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**Joint**

class GCGL.Baselines.jointtrain_model.NET(*args: Any, **kwargs: Any)

Jointly trained model baseline for GCGL tasks

**Parameters**

• **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
• **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

**observe**(data_loader, loss_criterion, task_i, args)

The method for learning the given tasks under the task-IL setting with multi-label classification datasets.

**Parameters**

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**observe_clsIL**(data_loader, loss_criterion, task_i, args)

The method for learning the given tasks under the class-IL setting with multi-class classification datasets.

**Parameters**

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**observe_tskIL_multicls**(train_loader_joint, loss_criterion, task_i, args)

The method for learning the given tasks under the task-IL setting with multi-class classification datasets.

**Parameters**

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**LwF**

**class** GCGL.Baselines.lwf_model.NET(*args: Any, **kwargs: Any)

LwF baseline for GCGL tasks

**Parameters**

• **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
• **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

**observe**(data_loader, loss_criterion, task_i, args, prev_model)

The method for learning the given tasks under the task-IL setting with multi-label classification datasets.

**Parameters**

• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().
• **prev_model** – The model obtained after learning the previous task.
observe_clsIL\( \text{\texttt{data\_loader, loss\_criterion, task\_i, args, prev\_model}} \)\)

The method for learning the given tasks under the class-IL setting with multi-class classification datasets.

Parameters
- \texttt{data\_loader} – The data loader for mini-batch training.
- \texttt{loss\_criterion} – The loss function.
- \texttt{task\_i} – Index of the current task.
- \texttt{args} – Same as the \texttt{args} in \texttt{\_init\_}().
- \texttt{prev\_model} – The model obtained after learning the previous task.

observe_tskIL\_multicls\( \text{\texttt{data\_loader, loss\_criterion, task\_i, args, prev\_model}} \)\)

The method for learning the given tasks under the task-IL setting with multi-class classification datasets.

Parameters
- \texttt{data\_loader} – The data loader for mini-batch training.
- \texttt{loss\_criterion} – The loss function.
- \texttt{task\_i} – Index of the current task.
- \texttt{args} – Same as the \texttt{args} in \texttt{\_init\_}().
- \texttt{prev\_model} – The model obtained after learning the previous task.

MAS

class \texttt{GCGL.Baselines.mas\_model.NET(*args: Any, **kwargs: Any)}\)

MAS baseline for GCGL tasks

Parameters
- \texttt{model} – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
- \texttt{args} – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

observe\( \text{\texttt{data\_loader, loss\_criterion, task\_i, args}}} \)\)

The method for learning the given tasks under the task-IL setting with multi-label classification datasets.

Parameters
- \texttt{data\_loader} – The data loader for mini-batch training.
- \texttt{loss\_criterion} – The loss function.
- \texttt{task\_i} – Index of the current task.
- \texttt{args} – Same as the \texttt{args} in \texttt{\_init\_}().

observe_clsIL\( \text{\texttt{data\_loader, loss\_criterion, task\_i, args}}} \)\)

The method for learning the given tasks under the class-IL setting with multi-class classification datasets.

Parameters
- \texttt{data\_loader} – The data loader for mini-batch training.
- \texttt{loss\_criterion} – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**observe_tskIL_multicls**(*data_loader, loss_criterion, task_i, args*)
The method for learning the given tasks under the task-IL setting with multi-class classification datasets.

**Parameters**
• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**TWP**

**class** GCGL.Baselines.twp_model.NET(*args: Any, **kwargs: Any*)
TWP baseline for GCGL tasks

**Parameters**
• **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
• **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

**observe**(*data_loader, loss_criterion, task_i, args*)
The method for learning the given tasks under the task-IL setting with multi-label classification datasets.

**Parameters**
• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**observe_clsIL**(*data_loader, loss_criterion, task_i, args*)
The method for learning the given tasks under the class-IL setting with multi-class classification datasets.

**Parameters**
• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
• **task_i** – Index of the current task.
• **args** – Same as the args in __init__().

**observe_tskIL_multicls**(*data_loader, loss_criterion, task_i, args*)
The method for learning the given tasks under the task-IL setting with multi-class classification datasets.

**Parameters**
• **data_loader** – The data loader for mini-batch training.
• **loss_criterion** – The loss function.
- **task_i** – Index of the current task.
- **args** – Same as the args in __init__().

### 1.1.3 Utility Functions

**NCGL.visualize.show_final_APAF** *(result_path, GCGL=False)*

The function to show the final AP and AF. Output are organized in a LaTeX firendly way.

- **Parameters**
  - **result_path** – The path to the experimental result

**NCGL.visualize.show_learning_curve** *(result_path, save_fig_name=None)*

The function to visualize the dynamics of AP.

- **Parameters**
  - **result_path** – The path to the experimental result
  - **save_fig_name** – If specified, the generated visualization will be stored with the specified name under the directory “./results/figures”

**NCGL.visualize.show_performance_matrices** *(result_path, save_fig_name=None, multiplier=1.0)*

The function to visualize the performance matrix.

- **Parameters**
  - **result_path** – The path to the experimental result
  - **save_fig_name** – If specified, the generated visualization will be stored with the specified name under the directory “./results/figures”

### 1.1.4 Implementations of New Methods

**class** **NCGL.Baselines.New_NCGL_model.NET** *(args: Any, **kwargs: Any)*

A template for implementing new methods for NCGL tasks. The major part for users to care about is the implementation of the function **observe()**, which is how the implemented NCGL method learns each new task.

- **Parameters**
  - **model** – The backbone GNNs, e.g. GCN, GAT, GIN, etc.
  - **task_manager** – Mainly serves to store the indices of the output dimensions corresponding to each task
  - **args** – The arguments containing the configurations of the experiments including the training parameters like the learning rate, the setting configurations like class-IL and task-IL, etc. These arguments are initialized in the train.py file and can be specified by the users upon running the code.

**observe**(args, g, features, labels, t, prev_model, train_ids, ids_per_cls, dataset)

The method for learning the given tasks. Each time a new task is presented, this function will be called to learn the task. Therefore, how the model adapts to new tasks and prevent forgetting on old tasks are all implemented in this function. More detailed comments accompanying the code can be found in the source code of this template in our GitHub repository.

- **Parameters**
  - **args** – Same as the args in __init__().
• **g** – The graph of the current task.
• **features** – Node features of the current task.
• **labels** – Labels of the nodes in the current task.
• **t** – Index of the current task.
• **prev_model** – The model obtained after learning the previous task.
• **train_ids** – The indices of the nodes participating in the training.
• **ids_per_cls** – Indices of the nodes in each class (not in use in the current baseline).
• **dataset** – The entire dataset (not in use in the current baseline).
g
GCGL.Baselines.bare_model, 17
GCGL.Baselines.ewc_model, 18
GCGL.Baselines.gem_model, 18
GCGL.Baselines.jointtrain_model, 19
GCGL.Baselines.lwf_model, 20
GCGL.Baselines.mas_model, 21
GCGL.Baselines.twp_model, 22

n
NCGL.Baselines.bare_model, 3
NCGL.Baselines.ergnn_model, 5
NCGL.Baselines.ewc_model, 6
NCGL.Baselines.gem_model, 8
NCGL.Baselines.joint_replay_all, 9
NCGL.Baselines.lwf_model, 12
NCGL.Baselines.mas_model, 13
NCGL.Baselines.New_NCGL_model, 23
NCGL.Baselines.twp_model, 15
NCGL.visualize, 23
INDEX

G
GCGL.Baselines.bare_model module, 17
GCGL.Baselines.ewc_model module, 18
GCGL.Baselines.gem_model module, 18
GCGL.Baselines.jointtrain_model module, 19
GCGL.Baselines.lwf_model module, 20
GCGL.Baselines.mas_model module, 21
GCGL.Baselines.twp_model module, 22

M
module
GCGL.Baselines.bare_model, 17
GCGL.Baselines.ewc_model, 18
GCGL.Baselines.gem_model, 18
GCGL.Baselines.jointtrain_model, 19
GCGL.Baselines.lwf_model, 20
GCGL.Baselines.mas_model, 21
GCGL.Baselines.twp_model, 22
NCGL.Baselines.bare_model, 3
NCGL.Baselines.ergnn_model, 5
NCGL.Baselines.ewc_model, 6
NCGL.Baselines.gem_model, 8
NCGL.Baselines.joint_replay_all, 9
NCGL.Baselines.lwf_model, 12
NCGL.Baselines.mas_model, 13
NCGL.Baselines.New_NCGL_model, 23
NCGL.Baselines.twp_model, 15
NCGL.visualize module, 23

O
observe() (GCGL.Baselines.bare_model.NET method), 17
observe() (GCGL.Baselines.ewc_model.NET method), 18
observe() (GCGL.Baselines.gem_model.NET method), 19
observe() (GCGL.Baselines.jointtrain_model.NET method), 19

N
NCGL.Baselines.bare_model module, 3
NCGL.Baselines.ergnn_model module, 5
NCGL.Baselines.ewc_model module, 6
NCGL.Baselines.gem_model module, 8
NCGL.Baselines.joint_replay_all module, 9
NCGL.Baselines.lwf_model module, 12
NCGL.Baselines.mas_model module, 13
NCGL.Baselines.New_NCGL_model module, 23
NCGL.Baselines.twp_model module, 15
NCGL.visualize module, 23
observe() (GCGL.Baselines.lwf_model.NET method), 20
observe() (GCGL.Baselines.mas_model.NET method), 21
observe() (GCGL.Baselines.twp_model.NET method), 22
observe() (NCGL.Baselines.bare_model.NET method), 3
observe() (NCGL.Baselines.ergnn_model.NET method), 5
observe() (NCGL.Baselines.ewc_model.NET method), 6
observe() (NCGL.Baselines.gem_model.NET method), 8
observe() (NCGL.Baselines.joint_replay_all.NET method), 9
observe() (NCGL.Baselines.lwf_model.NET method), 12
observe() (NCGL.Baselines.mas_model.NET method), 13
observe() (NCGL.Baselines.New_NCGL_model.NET method), 23
observe_class_IL_batch() (NCGL.Baselines.bare_model.NET method), 3
observe_class_IL_batch() (NCGL.Baselines.joint_replay_all.NET method), 4
observe_class_IL_batch() (NCGL.Baselines.ergnn_model.NET method), 5
observe_class_IL_batch() (NCGL.Baselines.ewc_model.NET method), 6
observe_class_IL_batch() (NCGL.Baselines.gem_model.NET method), 7
observe_class_IL_batch() (NCGL.Baselines.joint_replay_all.NET method), 8
observe_task_IL() (NCGL.Baselines.bare_model.NET method), 4
observe_task_IL() (NCGL.Baselines.ergnn_model.NET method), 6
observe_task_IL() (NCGL.Baselines.ewc_model.NET method), 7
observe_task_IL() (NCGL.Baselines.gem_model.NET method), 9
observe_task_IL() (NCGL.Baselines.joint_replay_all.NET method), 10
observe_task_IL() (NCGL.Baselines.lwf_model.NET method), 12
observe_task_IL() (NCGL.Baselines.mas_model.NET method), 14
observe_task_IL() (NCGL.Baselines.twp_model.NET method), 16
observe_task_IL_batch() (NCGL.Baselines.bare_model.NET method), 4
observe_task_IL_batch() (NCGL.Baselines.ergnn_model.NET method), 6
observe_task_IL_batch() (NCGL.Baselines.ewc_model.NET method), 7
observe_task_IL_batch() (NCGL.Baselines.gem_model.NET method), 9
observe_task_IL_batch() (NCGL.Baselines.joint_replay_all.NET method), 11
observe_task_IL_batch() (NCGL.Baselines.lwf_model.NET method), 13
observe_task_IL_batch() (NCGL.Baselines.mas_model.NET method), 14
observe_task_IL_batch() (NCGL.Baselines.twp_model.NET method), 16
observe_task_IL_crsedge() (NCGL.Baselines.joint_replay_all.NET method), 11
observe_tskIL_multicls() (GCGL.Baselines.bare_model.NET method), 17
observe_tskIL_multicls() (GCGL.Baselines.ewc_model.NET method), 18
observe_tskIL_multicls() (GCGL.Baselines.gem_model.NET method), 19
observe_tskIL_multicls() (GCGL.Baselines.jointtrain_model.NET method), 20
observe_tskIL_multiclass()
    (GCGL.Baselines.lwf_model.NET method), 21
observe_tskIL_multiclass()
    (GCGL.Baselines.mas_model.NET method), 22
observe_tskIL_multiclass()
    (GCGL.Baselines.twp_model.NET method), 22

S
show_final_APAF() (in module NCGL.visualize), 23
show_learning_curve() (in module NCGL.visualize), 23
show_performance_matrices() (in module NCGL.visualize), 23