Graph Gym Tutorial 2

Gabriele Santin

https://gabrielesantin.github.io/

TABLE OF CONTENTS

01 A bit of background

Initial idea published as the study of the design space of GNN

First hosted on

- **Stanford Snap**: "Design Space for Graph Neural Networks"
  http://snap.stanford.edu/gnn-design/

- **GitHub**: https://github.com/snap-stanford/graphgym

[1] J. You, R. Ying, J. Leskovec, Design Space for Graph Neural Networks.. Neural Information Processing Systems (NeurIPS), 2020.
https://arxiv.org/abs/2011.08843

01 A bit of background

Now also integrated in PyG

- **Intro page:** "Managing experiments with GraphGym"
  https://pytorch-geometric.readthedocs.io/en/latest/notes/graphgym.html
- **Docs:** https://pytorch-geometric.readthedocs.io/en/latest/modules/graphgym.html

01 A bit of background

We refer to this version in this tutorial

Now also integrated in PyG

- **Intro page:** "Managing experiments with GraphGym" https://pytorch-geometric.readthedocs.io/en/latest/notes/graphgym.html
- **Docs:** https://pytorch-geometric.readthedocs.io/en/latest/modules/graphgym.html

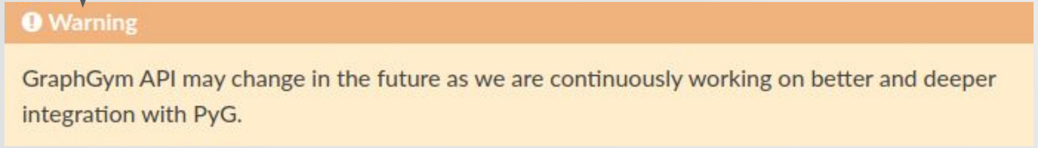
01 A bit of background

We refer to this version in this tutorial

Now also integrated in PyG

- **Intro page:** "Managing experiments with GraphGym" https://pytorch-geometric.readthedocs.io/en/latest/notes/graphgym.html
- **Docs:** https://pytorch-geometric.readthedocs.io/en/latest/modules/graphgym.html

Warning → Update to the last version of the packages


Warning

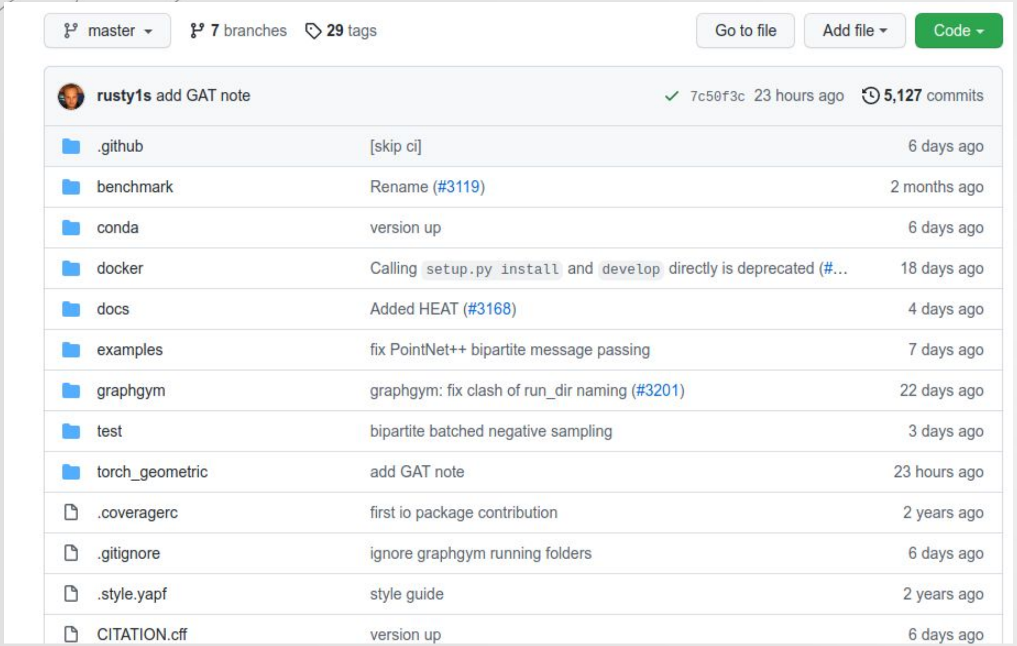
GraphGym API may change in the future as we are continuously working on better and deeper integration with PyG.

02 Goals and basic usage

Easy design and execution of experiments

- **Modularization:**
  - Pick a dataset, a model, a task, an evaluation metric, an optimizer
- **Reproducibility:**
  - Definition of simple configuration files to parametrize experiments
- **Scalability:**
  - Easy execution of multiple parallel experiments

02 Goals and basic usage

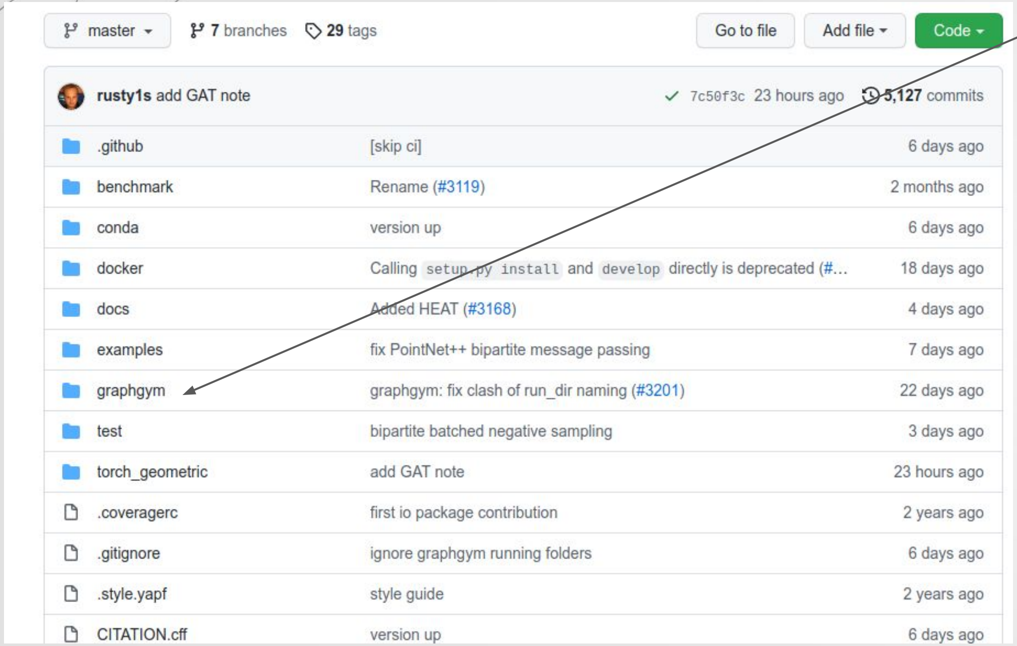


rusty1s add GAT note ✓ 7c50f3c 23 hours ago 5,127 commits

.github	[skip ci]	6 days ago
benchmark	Rename (#3119)	2 months ago
conda	version up	6 days ago
docker	Calling `setup.py install` and `develop` directly is deprecated (#...	18 days ago
docs	Added HEAT (#3168)	4 days ago
examples	fix PointNet++ bipartite message passing	7 days ago
graphgym	graphgym: fix clash of run_dir naming (#3201)	22 days ago
test	bipartite batched negative sampling	3 days ago
torch_geometric	add GAT note	23 hours ago
.coveragerc	first io package contribution	2 years ago
.gitignore	ignore graphgym running folders	6 days ago
.style.yapf	style guide	2 years ago
CITATION.cff	version up	6 days ago

02 Goals and basic usage

Copy/Clone



master	7 branches	29 tags

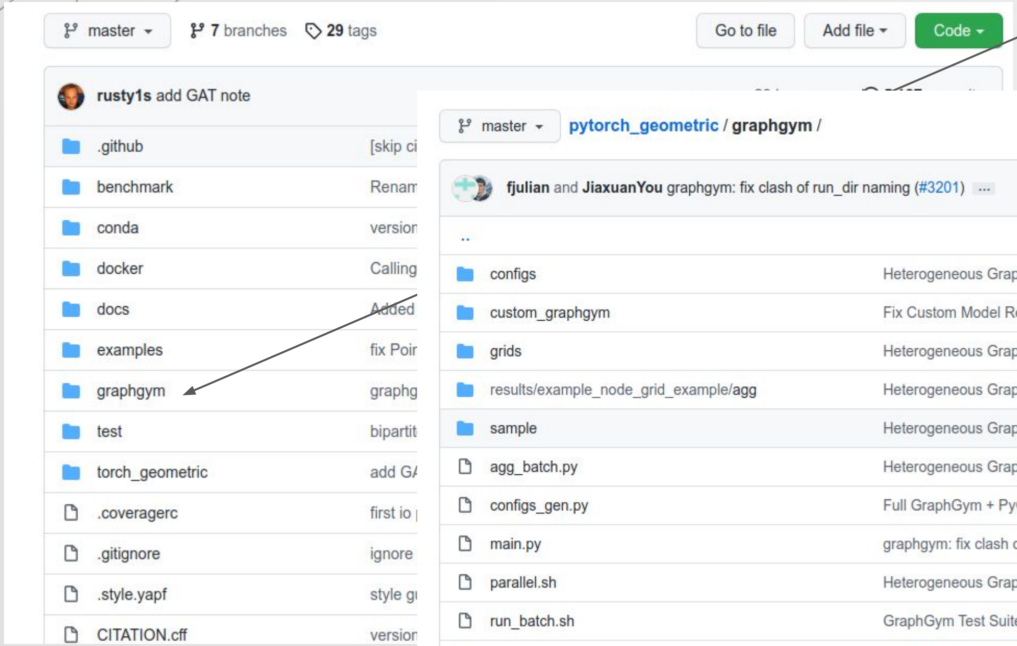
Go to file | Add file | Code

rusty1s add GAT note ✓ 7c50f3c 23 hours ago 5,127 commits

.github	[skip ci]	6 days ago
benchmark	Rename (#3119)	2 months ago
conda	version up	6 days ago
docker	Calling `setup.py install` and `develop` directly is deprecated (#...	18 days ago
docs	Added HEAT (#3168)	4 days ago
examples	fix PointNet++ bipartite message passing	7 days ago
graphgym	graphgym: fix clash of run_dir naming (#3201)	22 days ago
test	bipartite batched negative sampling	3 days ago
torch_geometric	add GAT note	23 hours ago
.coveragerc	first io package contribution	2 years ago
.gitignore	ignore graphgym running folders	6 days ago
.style.yapf	style guide	2 years ago
CITATION.cff	version up	6 days ago

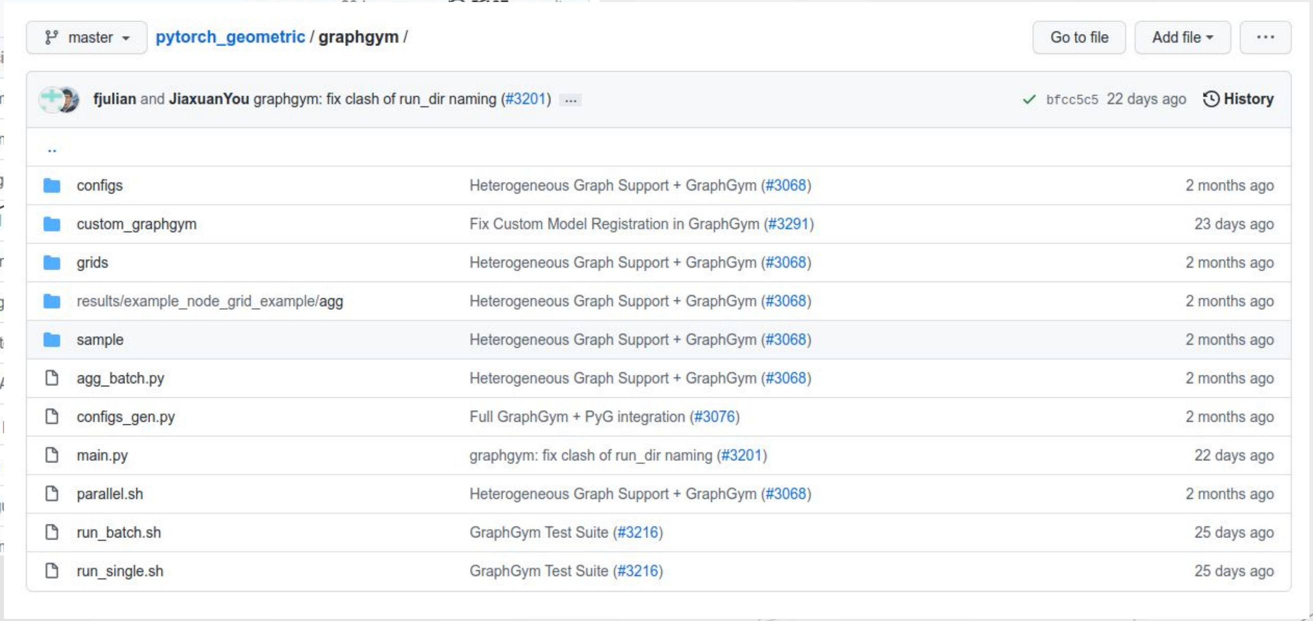
02 Goals and basic usage

Copy/Clone



rusty1s add GAT note

.github	[skip ci
benchmark	Renam
conda	version
docker	Calling
docs	Added
examples	fix Poir
graphgym	graphg
test	biparti
torch_geometric	add G/
.coveragerc	first io
.gitignore	ignore
.style.yapf	style g
CITATION.cff	version



pytorch_geometric / graphgym /

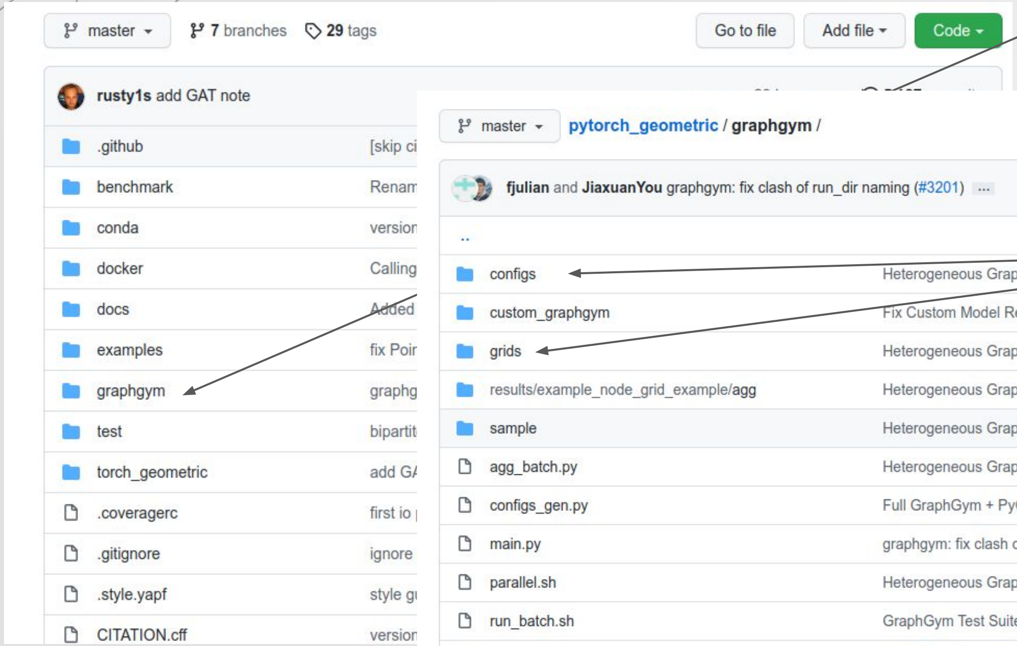
fjulian and JiaxuanYou graphgym: fix clash of run_dir naming (#3201) … ✓ bfcc5c5 22 days ago

..		
configs	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
custom_graphgym	Fix Custom Model Registration in GraphGym (#3291)	23 days ago
grids	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
results/example_node_grid_example/agg	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
sample	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
agg_batch.py	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
configs_gen.py	Full GraphGym + PyG integration (#3076)	2 months ago
main.py	graphgym: fix clash of run_dir naming (#3201)	22 days ago
parallel.sh	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
run_batch.sh	GraphGym Test Suite (#3216)	25 days ago
run_single.sh	GraphGym Test Suite (#3216)	25 days ago

02 Goals and basic usage

Copy/Clone

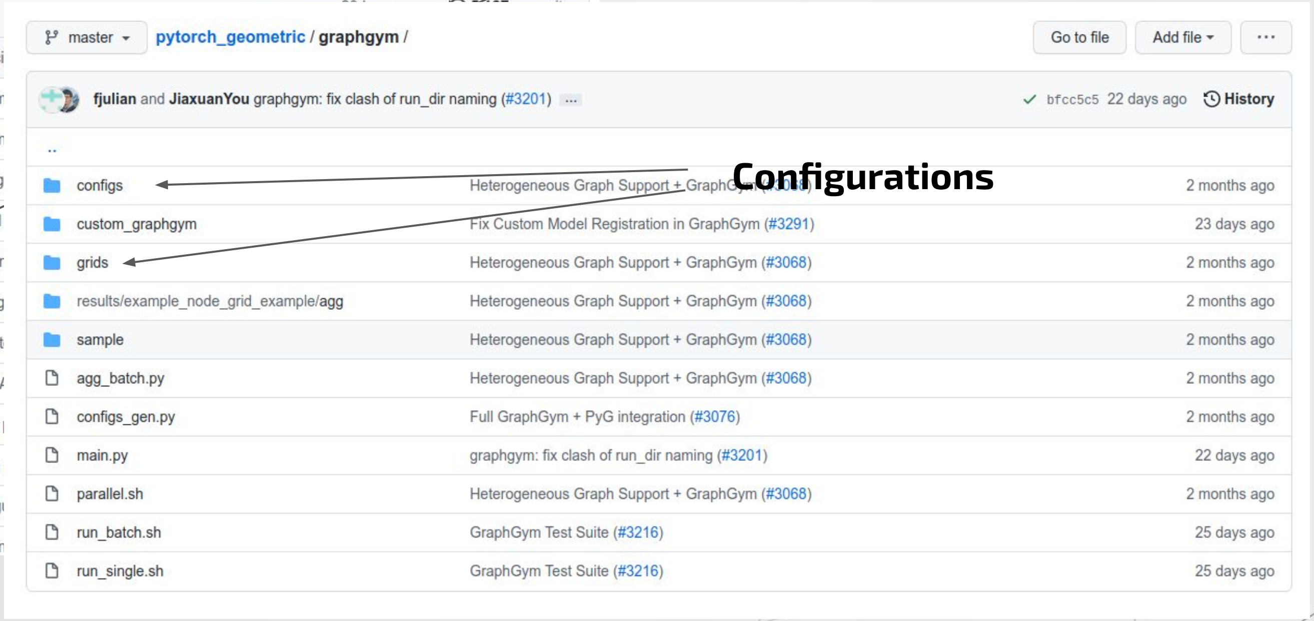
Configurations



master · 7 branches · 29 tags · Go to file · Add file · Code

rusty1s add GAT note

.github	[skip ci
benchmark	Renam
conda	version
docker	Calling
docs	Added
examples	fix Poir
graphgym	graphg
test	bipartit
torch_geometric	add G/
.coveragerc	first io
.gitignore	ignore
.style.yapf	style g
CITATION.cff	version

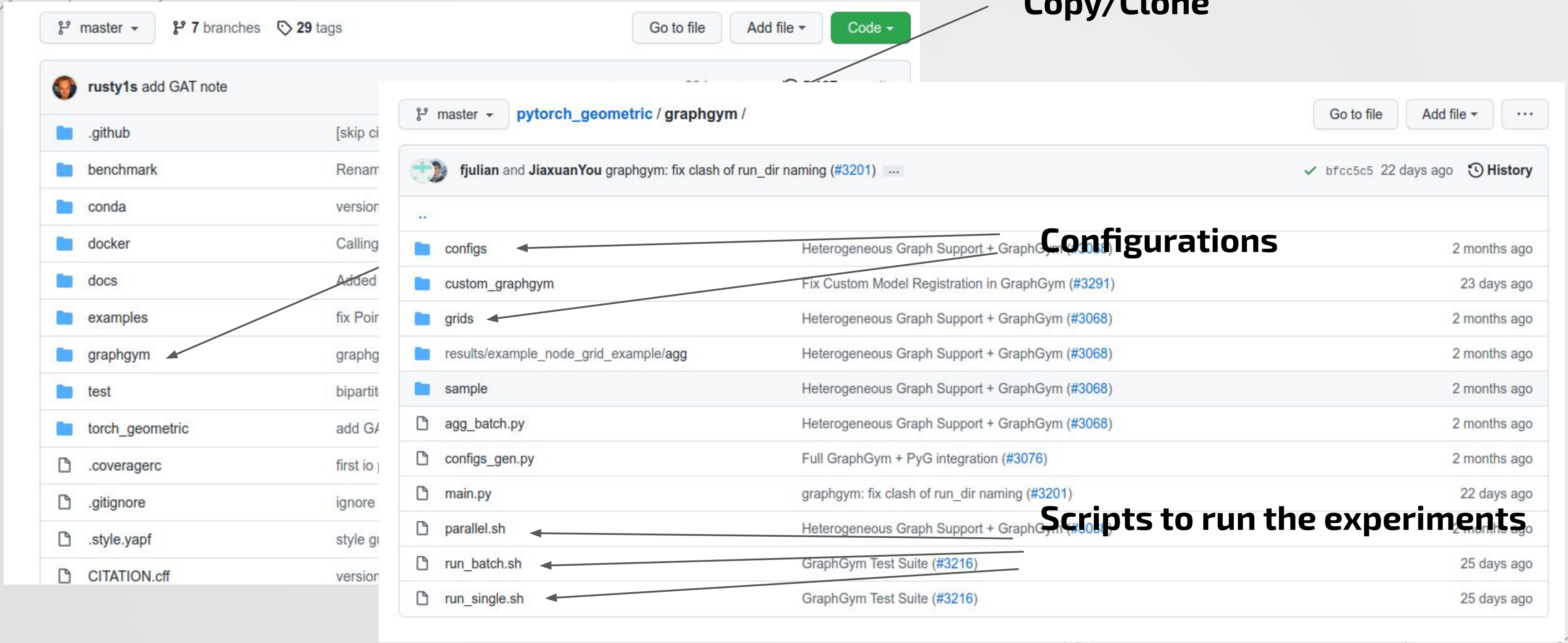


master · pytorch_geometric / graphgym / · Go to file · Add file · ...

fjulian and JiaxuanYou graphgym: fix clash of run_dir naming (#3201) ... ✓ bfcc5c5 22 days ago History

..		
configs	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
custom_graphgym	Fix Custom Model Registration in GraphGym (#3291)	23 days ago
grids	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
results/example_node_grid_example/agg	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
sample	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
agg_batch.py	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
configs_gen.py	Full GraphGym + PyG integration (#3076)	2 months ago
main.py	graphgym: fix clash of run_dir naming (#3201)	22 days ago
parallel.sh	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
run_batch.sh	GraphGym Test Suite (#3216)	25 days ago
run_single.sh	GraphGym Test Suite (#3216)	25 days ago

02 Goals and basic usage



master · 7 branches · 29 tags · Go to file · Add file · Code

rusty1s add GAT note

.github	[skip ci
benchmark	Renam
conda	version
docker	Calling
docs	Added
examples	fix Poir
graphgym	graphg
test	bipartit
torch_geometric	add GA
.coveragerc	first io
.gitignore	ignore
.style.yapf	style gu
CITATION.cff	version

master · pytorch_geometric / graphgym / · Go to file · Add file · ...

fjulian and JiaxuanYou graphgym: fix clash of run_dir naming (#3201) ... ✓ bfcc5c5 22 days ago · History

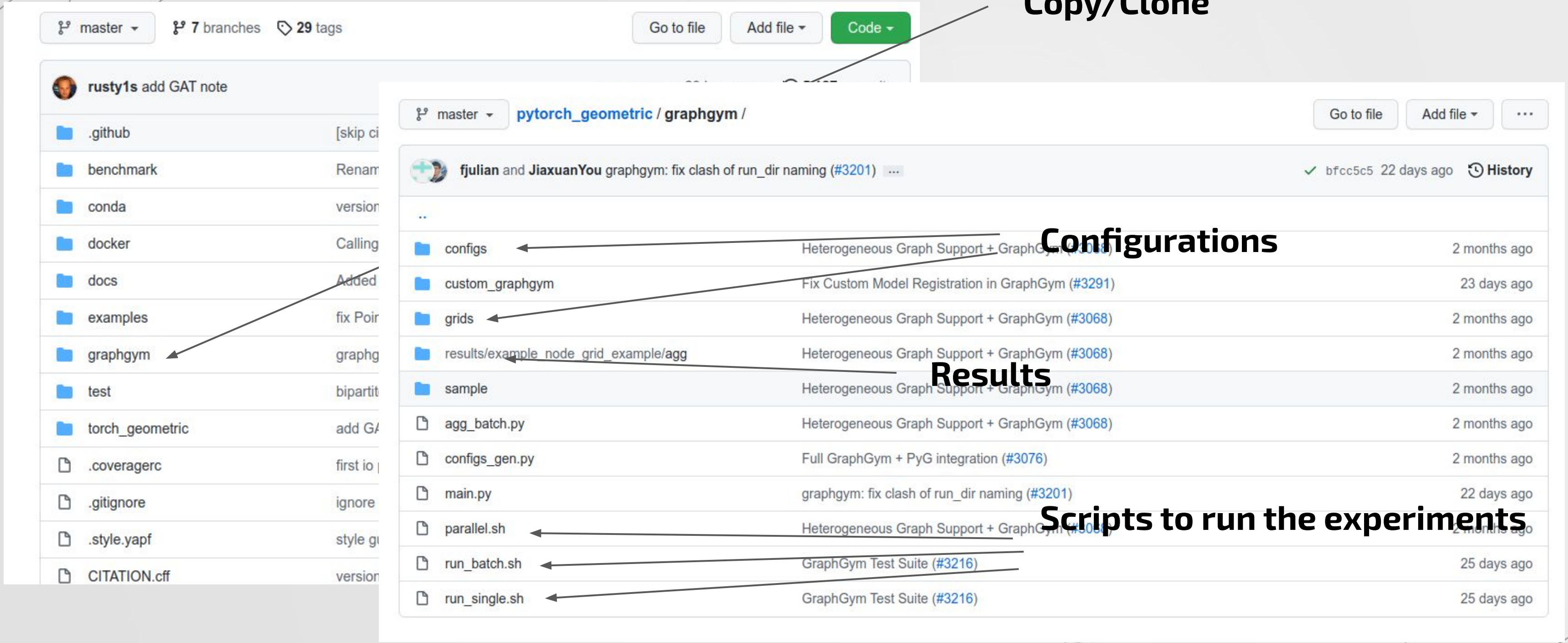
..		
configs	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
custom_graphgym	Fix Custom Model Registration in GraphGym (#3291)	23 days ago
grids	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
results/example_node_grid_example/agg	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
sample	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
agg_batch.py	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
configs_gen.py	Full GraphGym + PyG integration (#3076)	2 months ago
main.py	graphgym: fix clash of run_dir naming (#3201)	22 days ago
parallel.sh	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
run_batch.sh	GraphGym Test Suite (#3216)	25 days ago
run_single.sh	GraphGym Test Suite (#3216)	25 days ago

Copy/Clone

Configurations

Scripts to run the experiments

02 Goals and basic usage



master · 7 branches · 29 tags · Go to file · Add file · Code

rusty1s add GAT note

.github	[skip ci
benchmark	Renam
conda	version
docker	Calling
docs	Added
examples	fix Poir
graphgym	graphg
test	bipartit
torch_geometric	add G/
.coveragerc	first io
.gitignore	ignore
.style.yapf	style g
CITATION.cff	version

master · pytorch_geometric / graphgym / · Go to file · Add file · ···

fjulian and JiaxuanYou graphgym: fix clash of run_dir naming (#3201) ··· ✓ bfcc5c5 22 days ago · History

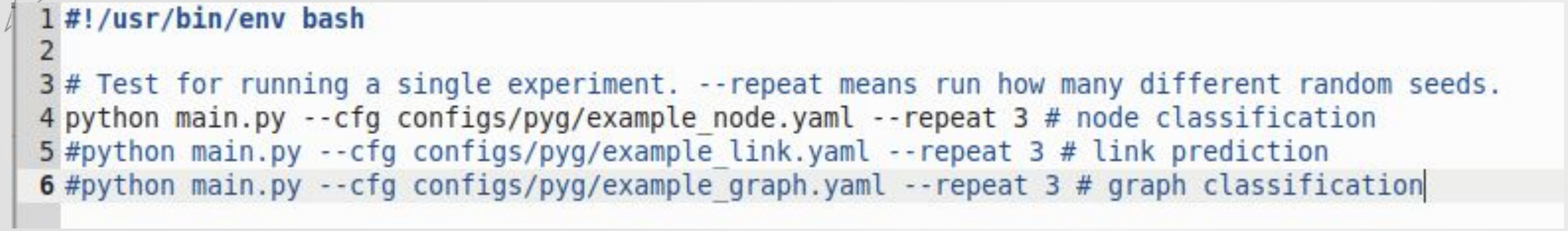
..		
configs	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
custom_graphgym	Fix Custom Model Registration in GraphGym (#3291)	23 days ago
grids	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
results/example_node_grid_example/agg	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
sample	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
agg_batch.py	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
configs_gen.py	Full GraphGym + PyG integration (#3076)	2 months ago
main.py	graphgym: fix clash of run_dir naming (#3201)	22 days ago
parallel.sh	Heterogeneous Graph Support + GraphGym (#3068)	2 months ago
run_batch.sh	GraphGym Test Suite (#3216)	25 days ago
run_single.sh	GraphGym Test Suite (#3216)	25 days ago

03 Example: node classification

Run run_single.sh . . .

03 Example: node classification

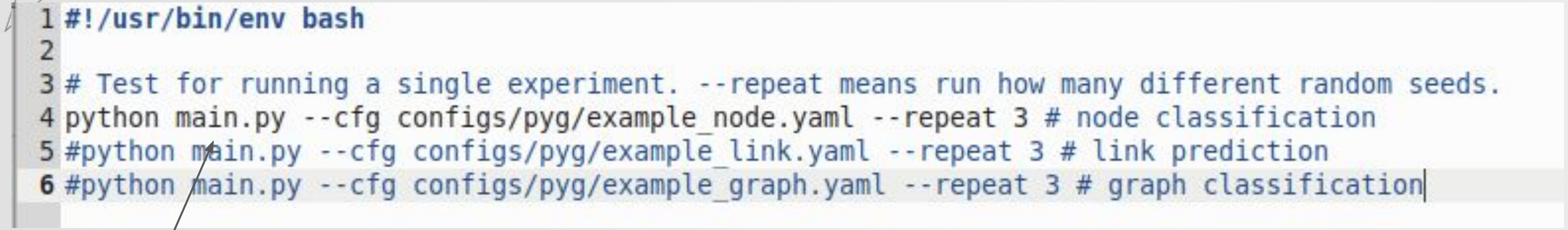
Run run_single.sh . . .



```
#!/usr/bin/env bash

# Test for running a single experiment. --repeat means run how many different random seeds.
python main.py --cfg configs/pyg/example_node.yaml --repeat 3 # node classification
#python main.py --cfg configs/pyg/example_link.yaml --repeat 3 # link prediction
#python main.py --cfg configs/pyg/example_graph.yaml --repeat 3 # graph classification
```

03 Example: node classification

Run run_single.sh . . .

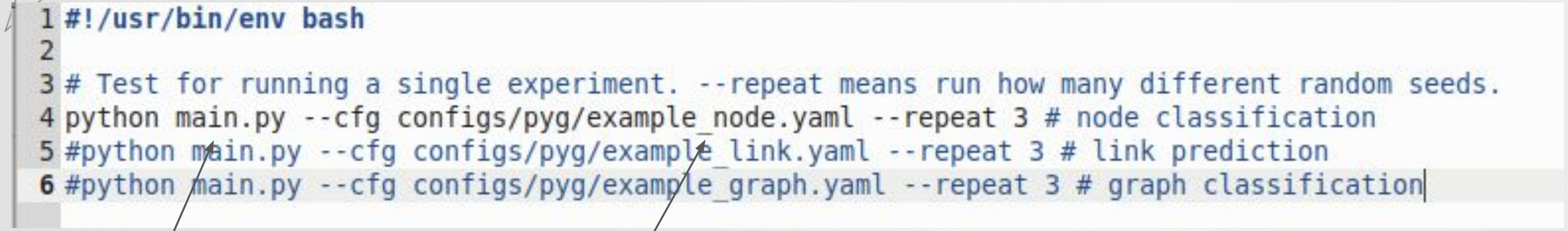


```
#!/usr/bin/env bash

# Test for running a single experiment. --repeat means run how many different random seeds.
python main.py --cfg configs/pyg/example_node.yaml --repeat 3 # node classification
#python main.py --cfg configs/pyg/example_link.yaml --repeat 3 # link prediction
#python main.py --cfg configs/pyg/example_graph.yaml --repeat 3 # graph classification
```

Base Graph Gym
parsing & execution

03 Example: node classification

Run run_single.sh . . .



```
#!/usr/bin/env bash

# Test for running a single experiment. --repeat means run how many different random seeds.
python main.py --cfg configs/pyg/example_node.yaml --repeat 3 # node classification
#python main.py --cfg configs/pyg/example_link.yaml --repeat 3 # link prediction
#python main.py --cfg configs/pyg/example_graph.yaml --repeat 3 # graph classification
```

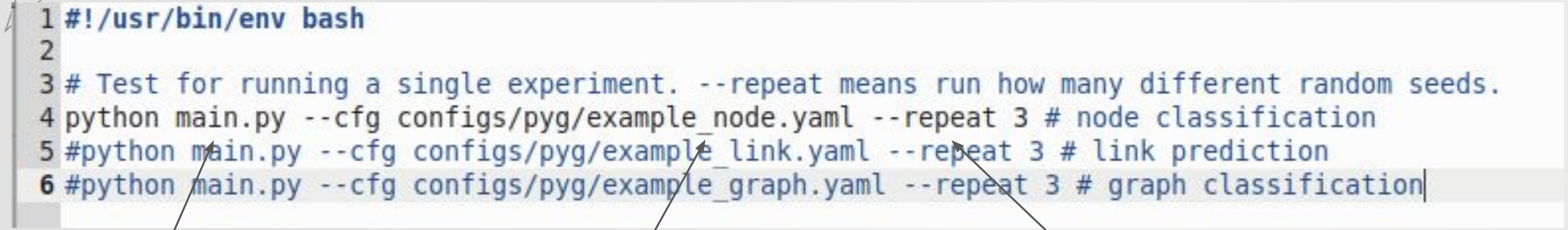
Configuration file

- define any aspect of the experiment

Base Graph Gym parsing & execution

03 Example: node classification

Run run_single.sh . . .



```bash
#!/usr/bin/env bash

# Test for running a single experiment. --repeat means run how many different random seeds.
python main.py --cfg configs/pyg/example_node.yaml --repeat 3 # node classification
#python main.py --cfg configs/pyg/example_link.yaml --repeat 3 # link prediction
#python main.py --cfg configs/pyg/example_graph.yaml --repeat 3 # graph classification
```

Base Graph Gym parsing & execution

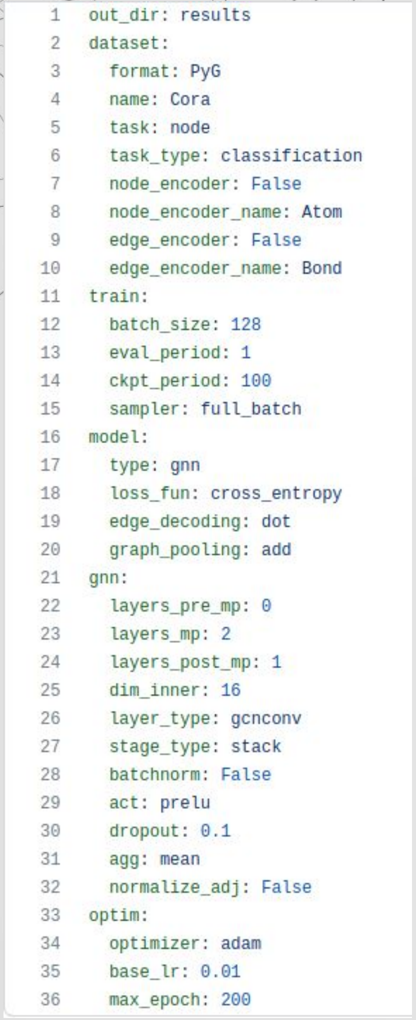
Configuration file

- define any aspect of the experiment

Number of **repetitions** of the single experiment

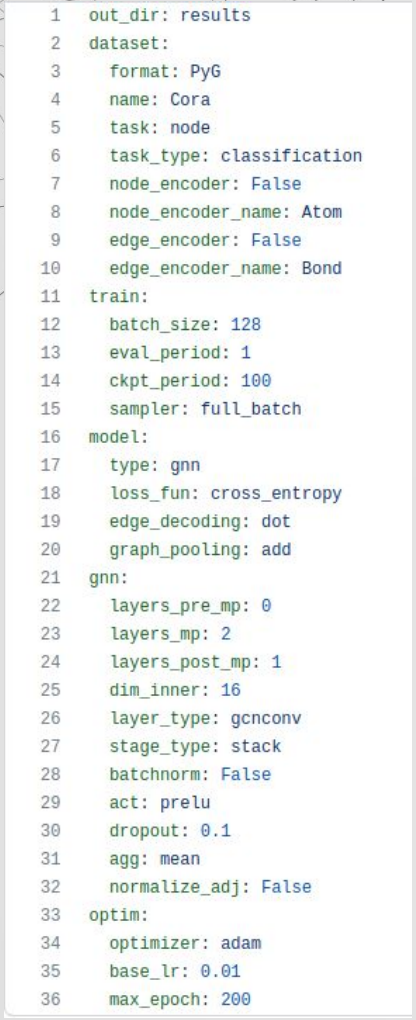
03 Example: node classification

This is configs/pyg/example_node.yaml



```yaml
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

03 Example: node classification

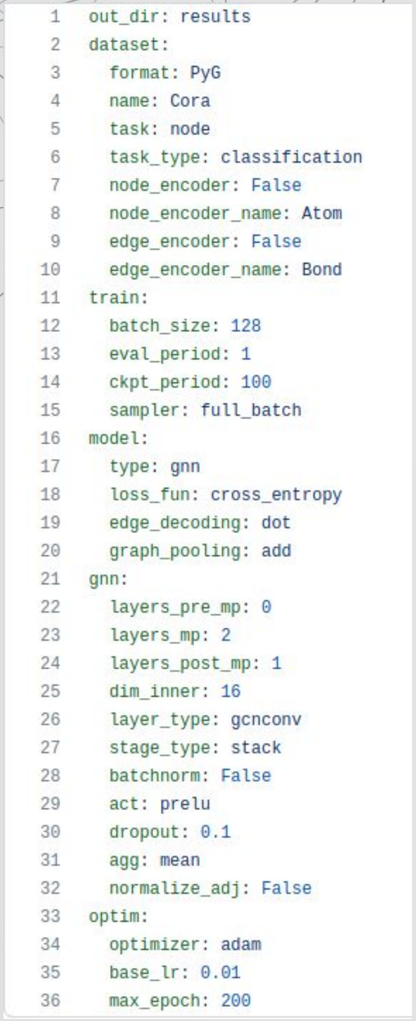


```yaml
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

This is configs/pyg/example_node.yaml

- It defines some **parameters** divided into groups
- **Unspecified** parameters are set to the default values
- A **list of all** parameters, with explanation and default values is in `torch_geometric.graphgym.set_cfg()` https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/graphgym/config.html#set_cfg

03 Example: node classification



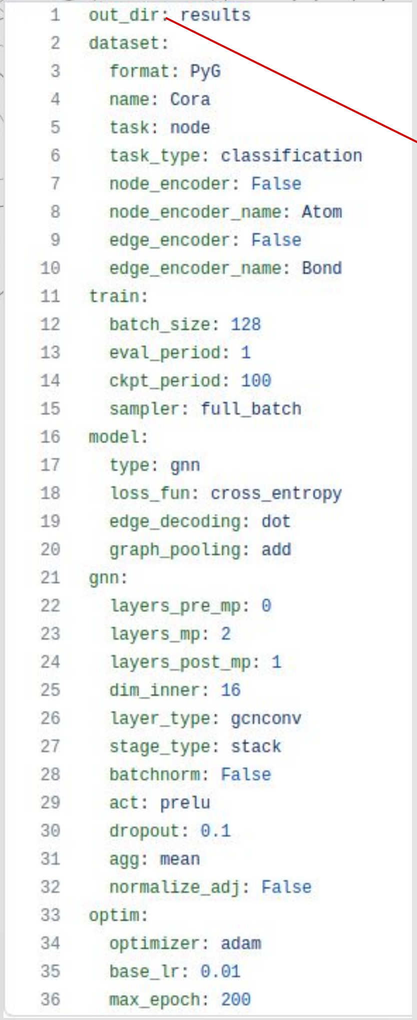
```yaml
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

This is configs/pyg/example_node.yaml

- It defines some **parameters** divided into groups
- **Unspecified** parameters are set to the default values
- A **list of all** parameters, with explanation and default values is in `torch_geometric.graphgym.set_cfg()` https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/graphgym/config.html#set_cfg

Warning: In this example some param seem to be useless

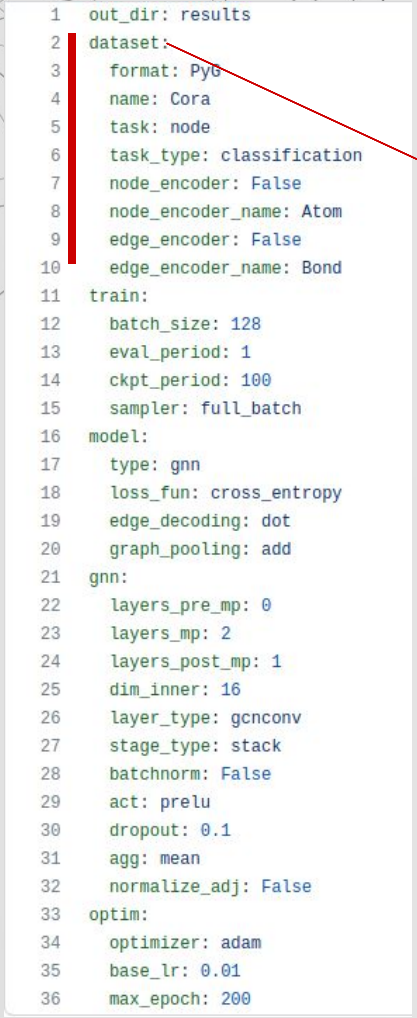
03 Example: node classification



```
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

Folder to save the results

03 Example: node classification

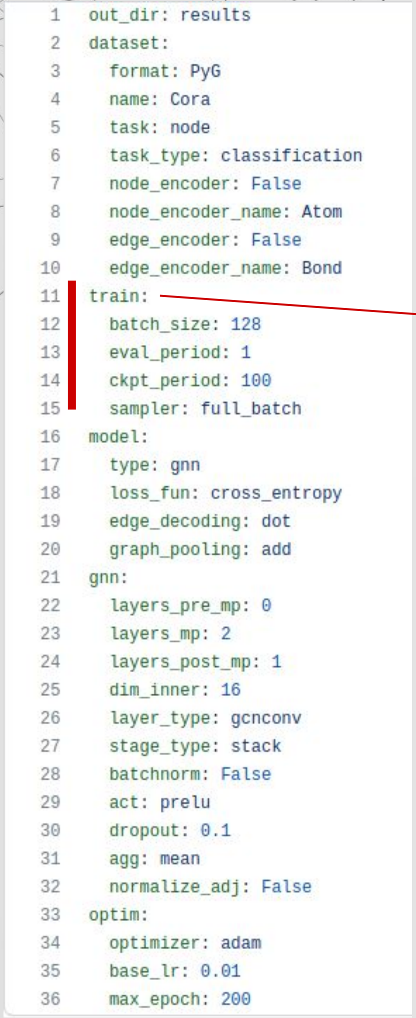


```yaml
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

Dataset and task specification

- **format**: PyG, NetworkX
- **name**
- **task**: node, edge, graph, link_pred
- **task_type**: classification, regression, classification_binary
- **node_encoder** / **edge_encoder**: bool, use encoder for node / edge features
- **node_encoder_name**, **edge_encoder_name**

03 Example: node classification

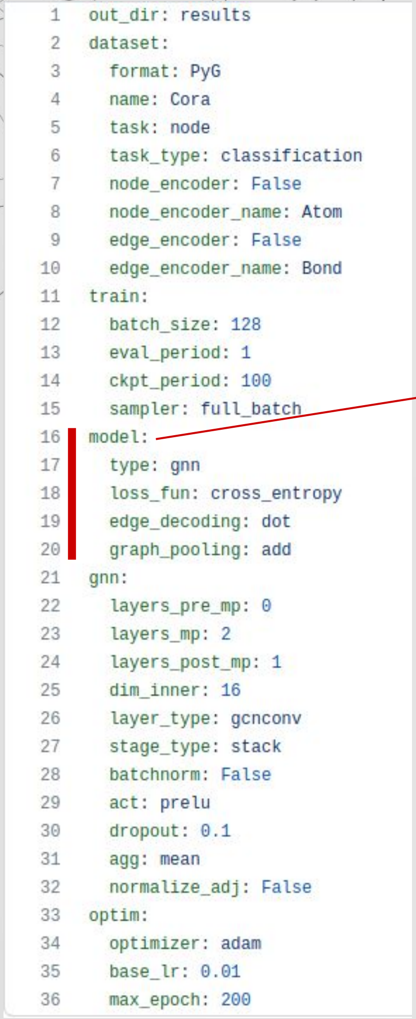


```
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

Training parameters

- **batch_size**: graph mini-batch size
- **eval_period:** evaluate on test every # epochs
- **ckpt_period**: save checkpoint every # epochs
- **sampler**: sampling strategy for the train loader

03 Example: node classification

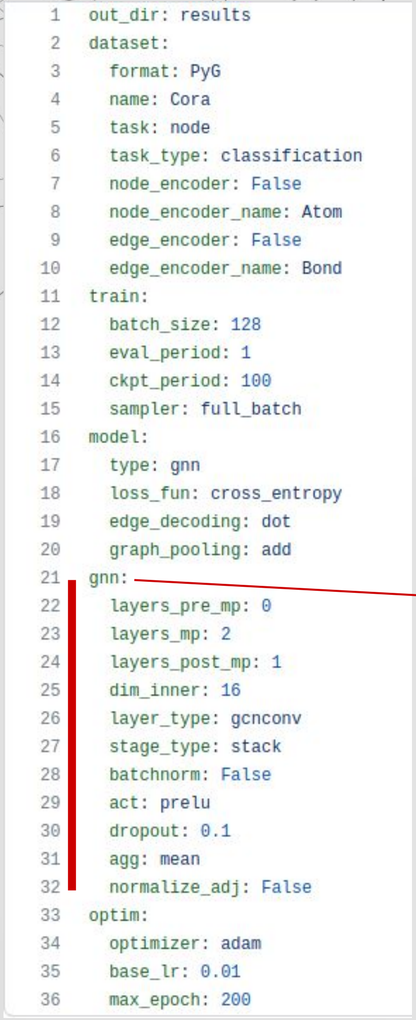


```
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

Model specification

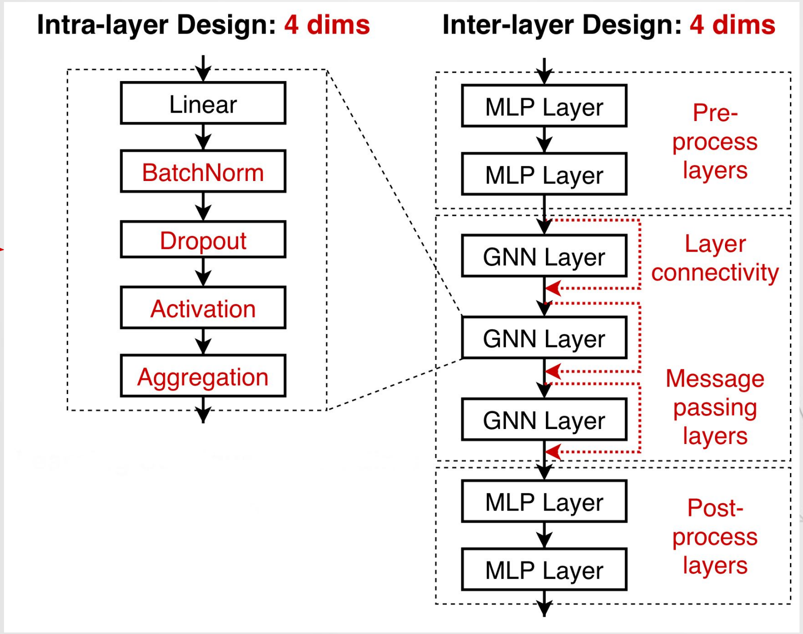
- **type:** gnn
- **loss_fun:** cross_entropy, mse
- **edge_decoding**: dot, cosine_similarity, concat
- **graph_pooling**: add, mean, max

03 Example: node classification

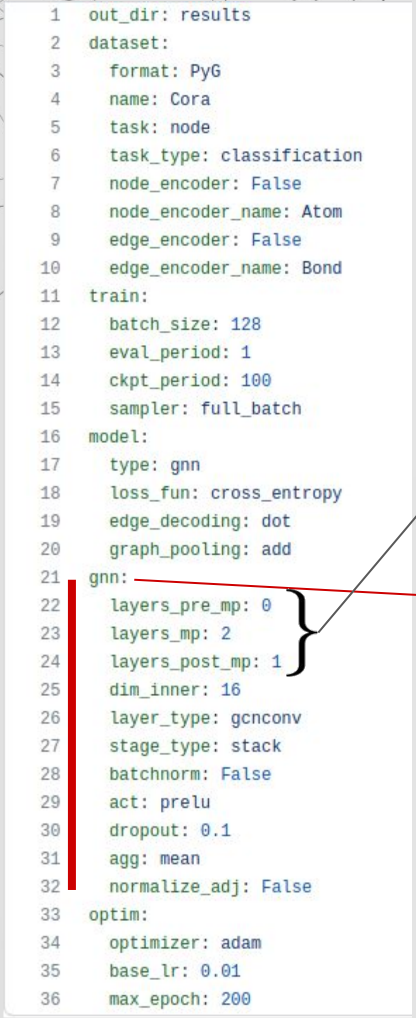


```
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

GNN architecture specification



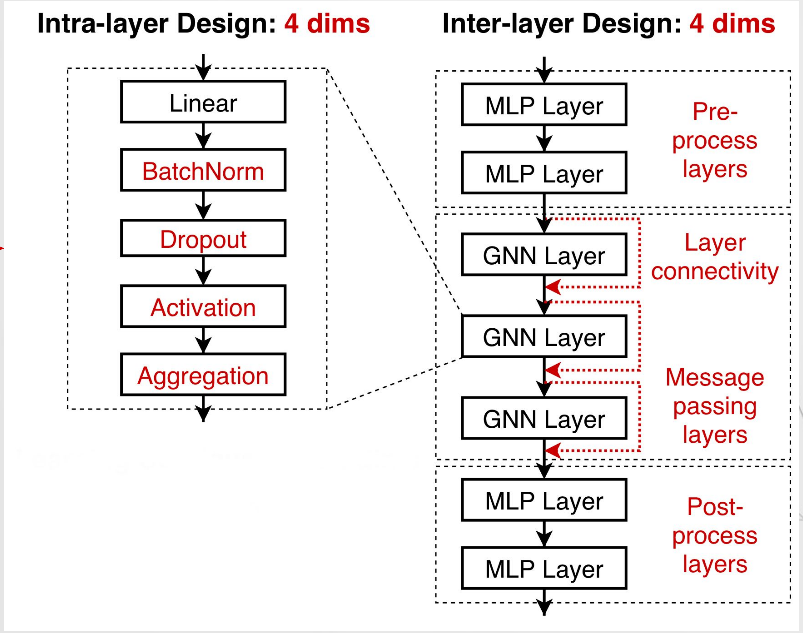
03 Example: node classification



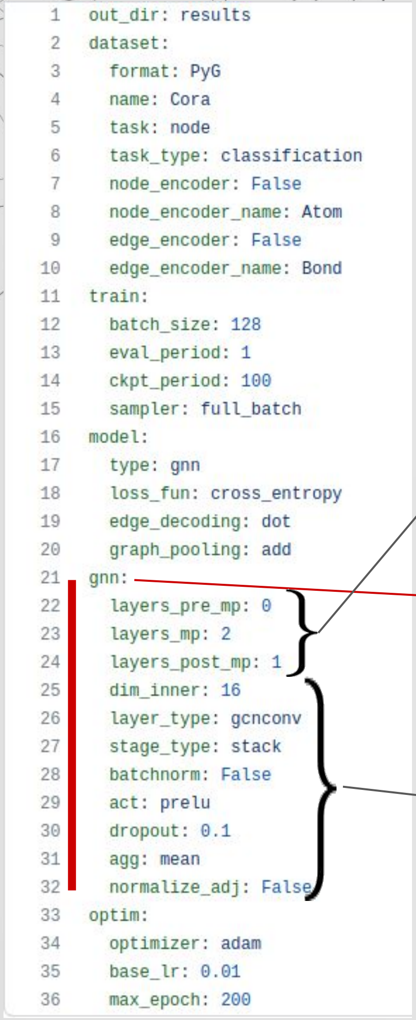
```
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

Number of layers pre/GNN/post

GNN architecture specification



03 Example: node classification

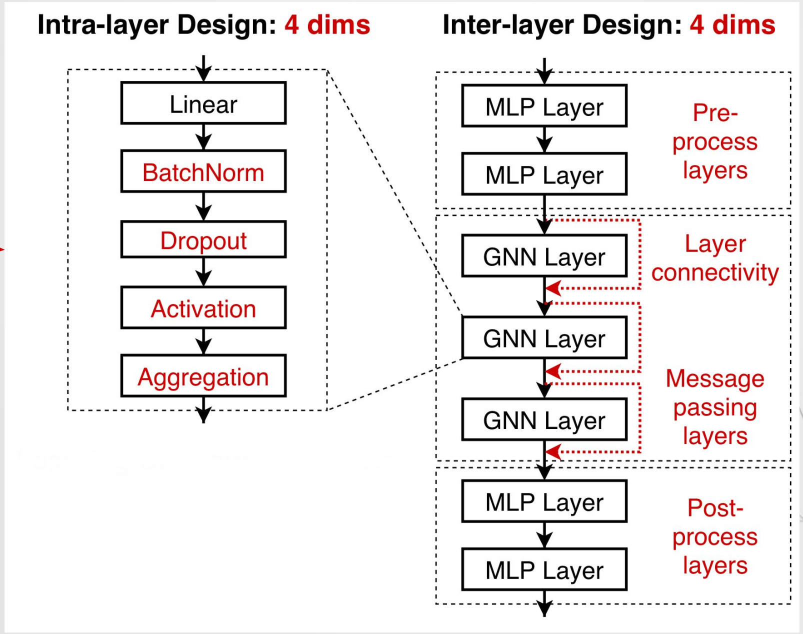


```
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

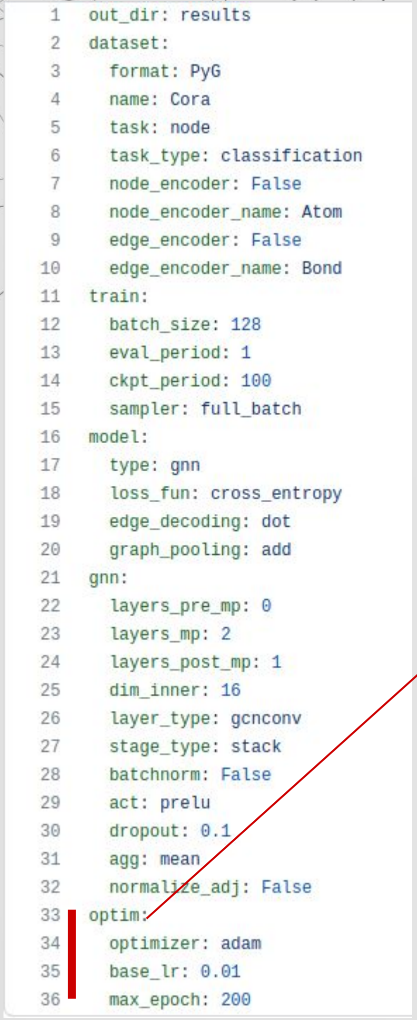
Number of layers pre/GNN/post

GNN options

GNN architecture specification



03 Example: node classification


```
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

Optimization options

- **optimizer:** sgd, adam **CHECK**
- **base_lr**
- **max_epoch**

03 Example: node classification

A bit more work to find all the options:

03 Example: node classification

A bit more work to find all the options:

1. A **list of all** parameters, with explanation and default values is in `torch_geometric.graphgym.set_cfg()`
   - https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/graphgym/config.html#set_cfg

2. See also the single modules
   - https://pytorch-geometric.readthedocs.io/en/latest/_modules/index.html
   - https://github.com/pyg-team/pytorch_geometric/tree/master/torch_geometric/graphgym

03 Example: node classification

A bit more work to find all the options:

1. A **list of all** parameters, with explanation and default values is in `torch_geometric.graphgym.set_cfg()`
    - https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/graphgym/config.html#set_cfg

2. See also the single modules
    - https://pytorch-geometric.readthedocs.io/en/latest/_modules/index.html
    - https://github.com/pyg-team/pytorch_geometric/tree/master/torch_geometric/graphgym

There is no automated way to ensure that every option that is available in 2) is also listed in 1)

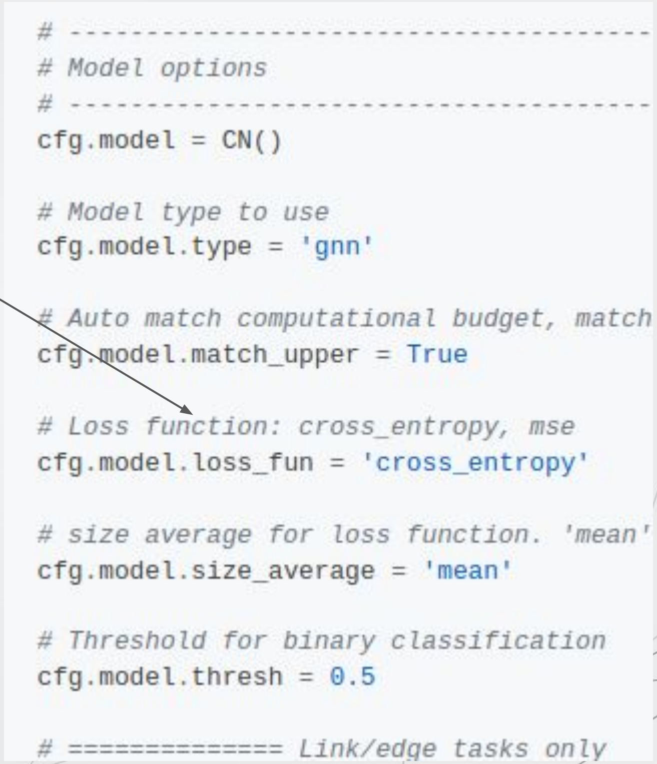
03 Example: node classification

A bit more work to find all the options:

1. A **list of all** parameters, with explanation and default values is in `torch_geometric.graphgym.set_cfg()`
   - https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/graphgym/config.html#set_cfg

2. See also the single modules
   - https://pytorch-geometric.readthedocs.io/en/latest/_modules/index.html
   - https://github.com/pyg-team/pytorch_geometric/tree/master/torch_geometric/graphgym

There is no automated way to ensure that every option that is available in 2) is also listed in 1)



```
# ----------------------------------------
# Model options
# ----------------------------------------
cfg.model = CN()

# Model type to use
cfg.model.type = 'gnn'

# Auto match computational budget, match
cfg.model.match_upper = True

# Loss function: cross_entropy, mse
cfg.model.loss_fun = 'cross_entropy'

# size average for loss function. 'mean'
cfg.model.size_average = 'mean'

# Threshold for binary classification
cfg.model.thresh = 0.5

# ============== Link/edge tasks only
```

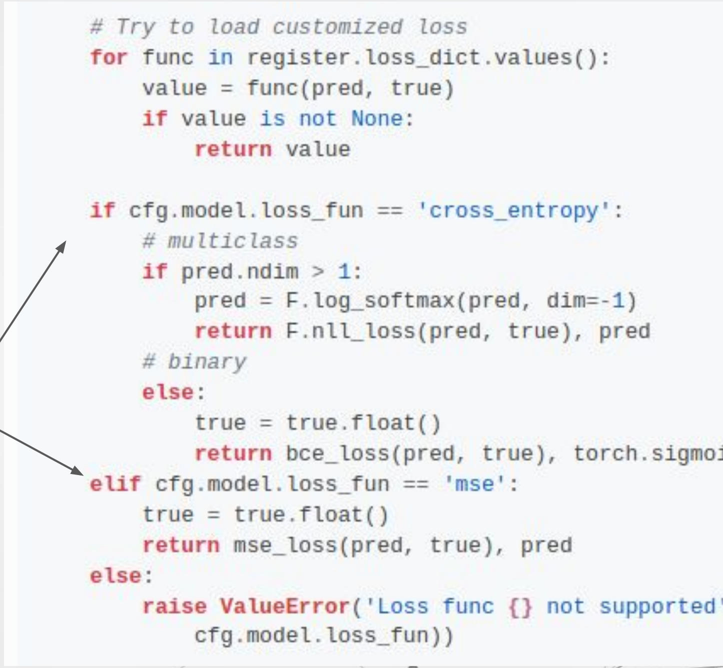
03 Example: node classification

A bit more work to find all the options:

1. A **list of all** parameters, with explanation and default values is in `torch_geometric.graphgym.set_cfg()`
   - https://pytorch-geometric.readthedocs.io/en/latest/_modules/torch_geometric/graphgym/config.html#set_cfg

2. See also the single modules
   - https://pytorch-geometric.readthedocs.io/en/latest/_modules/index.html
   - https://github.com/pyg-team/pytorch_geometric/tree/master/torch_geometric/graphgym

There is no automated way to ensure that every option that is available in 2) is also listed in 1)



```
# Try to load customized loss
for func in register.loss_dict.values():
    value = func(pred, true)
    if value is not None:
        return value

if cfg.model.loss_fun == 'cross_entropy':
    # multiclass
    if pred.ndim > 1:
        pred = F.log_softmax(pred, dim=-1)
        return F.nll_loss(pred, true), pred
    # binary
    else:
        true = true.float()
        return bce_loss(pred, true), torch.sigmoi
elif cfg.model.loss_fun == 'mse':
    true = true.float()
    return mse_loss(pred, true), pred
else:
    raise ValueError('Loss func {} not supported'
        cfg.model.loss_fun))
```

torch_geometric.graphgym.loss

03 Example: node classification

Results written in results/${CONFIG_NAME}/

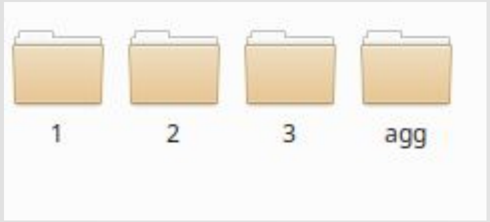
results/example_node/

03 Example: node classification

Results written in results/${CONFIG_NAME}/

$\uparrow$

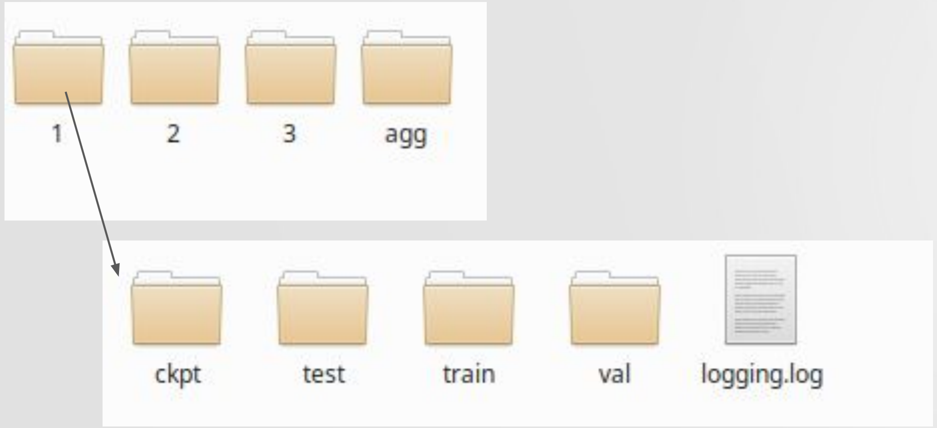
results/example_node/



03 Example: node classification

Results written in results/${CONFIG_NAME}/

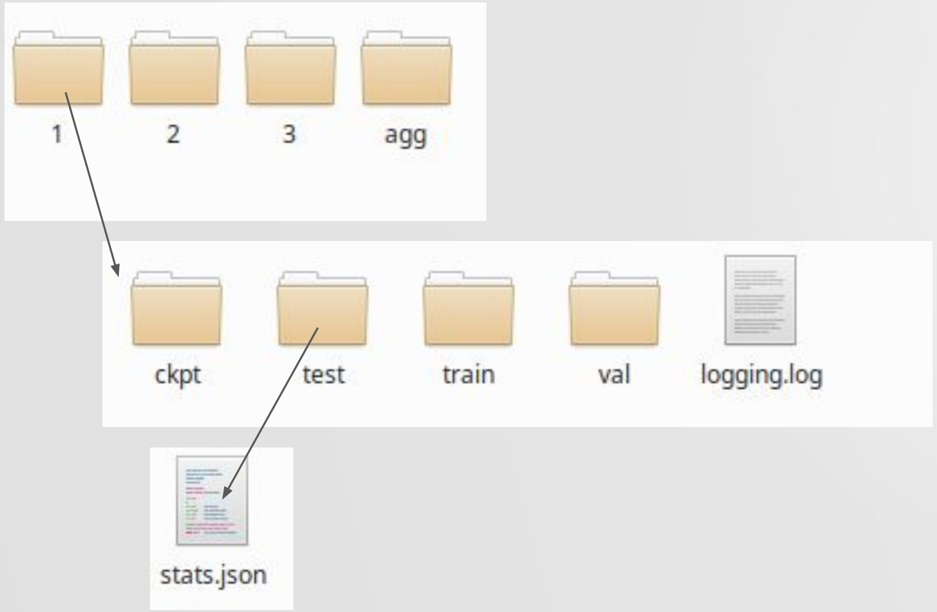
results/example_node/



03 Example: node classification

Results written in results/${CONFIG_NAME}/

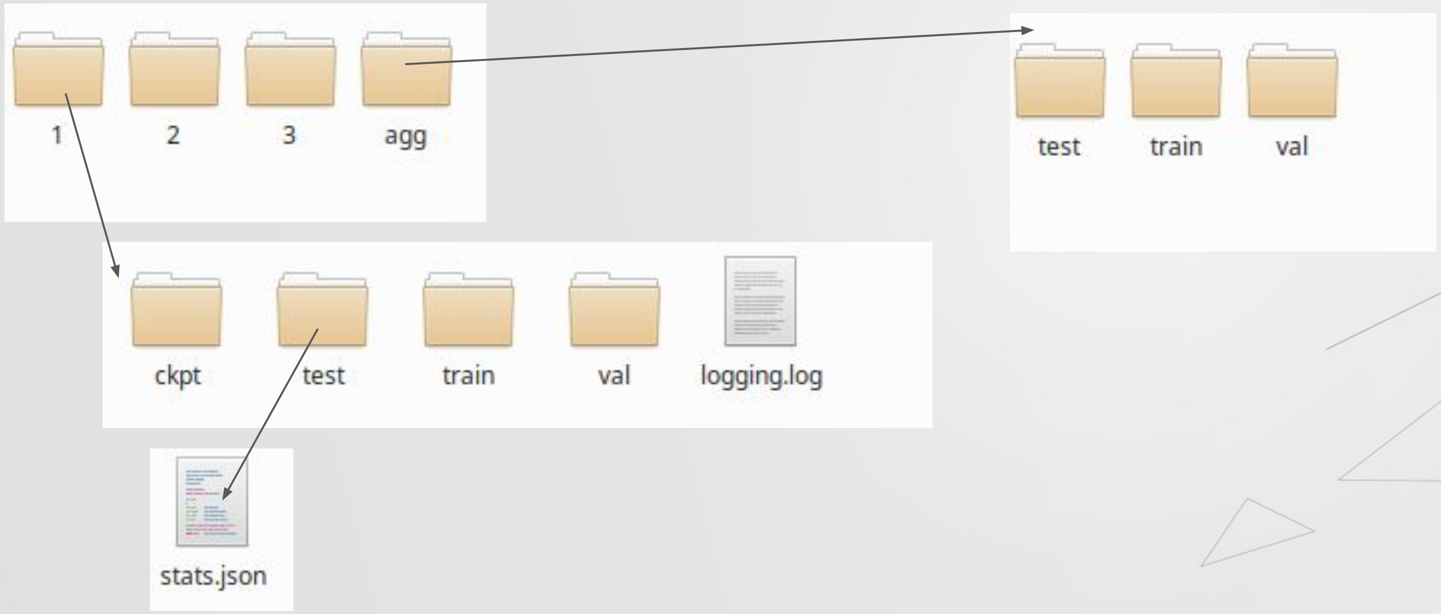
results/example_node/



03 Example: node classification

Results written in results/${CONFIG_NAME}/

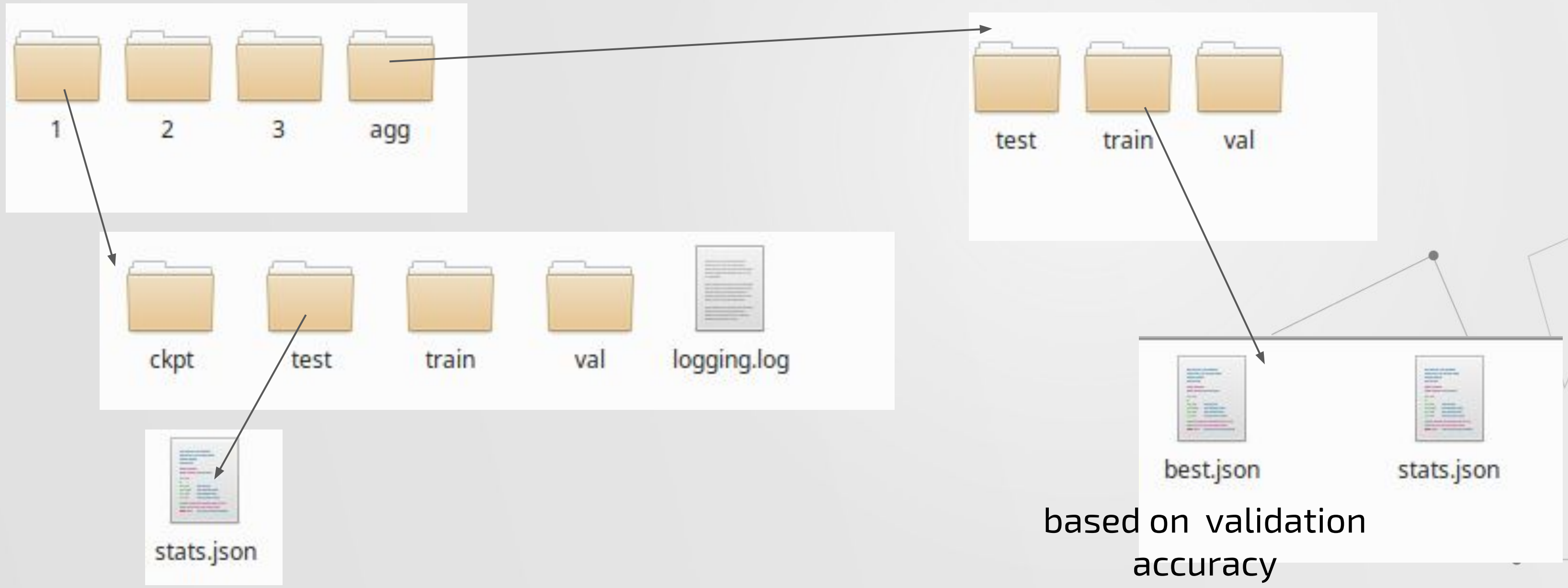
results/example_node/



03 Example: node classification

Results written in results/${CONFIG_NAME}/

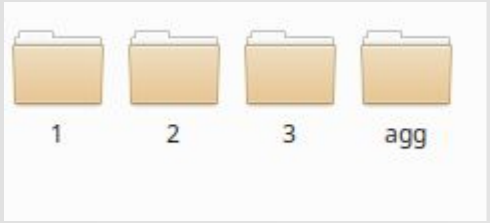
results/example_node/



03 Example: node classification

Results written in results/${CONFIG_NAME}/

results/example_node/



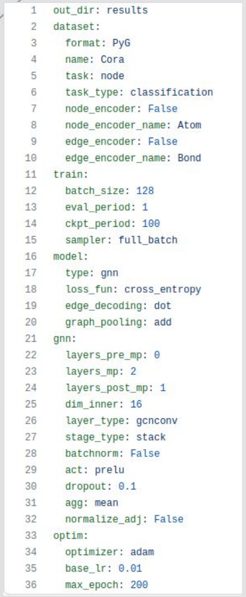
Let's **check** this . . .

04 Example: batch training

Idea define multiple experiments by perturbing a base config file

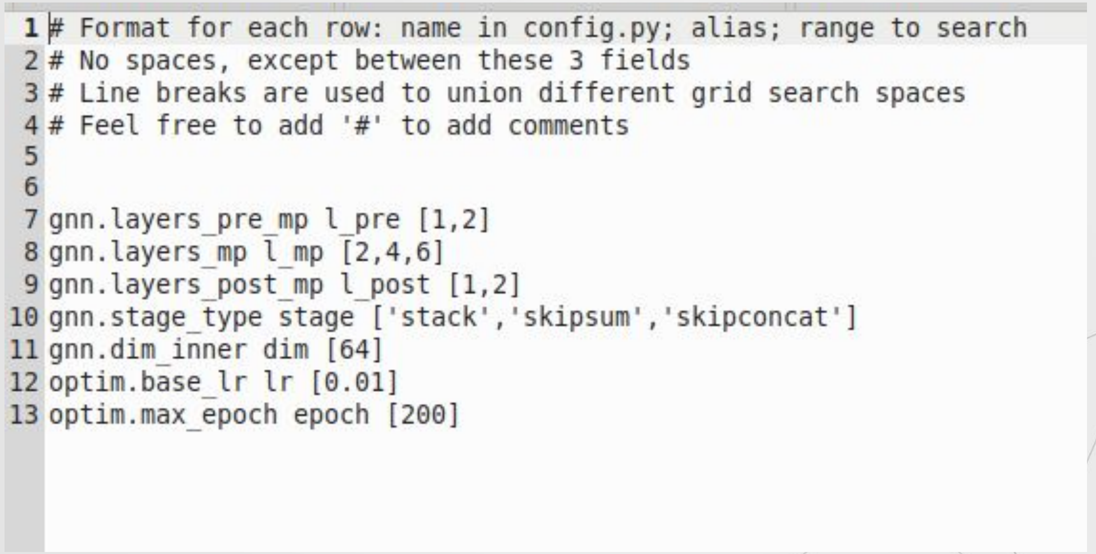
04 Example: batch training

Idea define multiple experiments by perturbing a base config file



```yaml
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

+



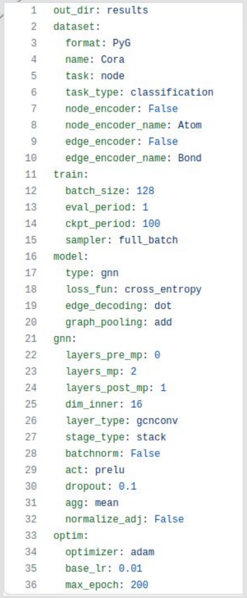
```
# Format for each row: name in config.py; alias; range to search
# No spaces, except between these 3 fields
# Line breaks are used to union different grid search spaces
# Feel free to add '#' to add comments


gnn.layers_pre_mp l_pre [1,2]
gnn.layers_mp l_mp [2,4,6]
gnn.layers_post_mp l_post [1,2]
gnn.stage_type stage ['stack','skipsum','skipconcat']
gnn.dim_inner dim [64]
optim.base_lr lr [0.01]
optim.max_epoch epoch [200]
```

configs/pyg/example_node.yaml

grids/pyg/example.txt

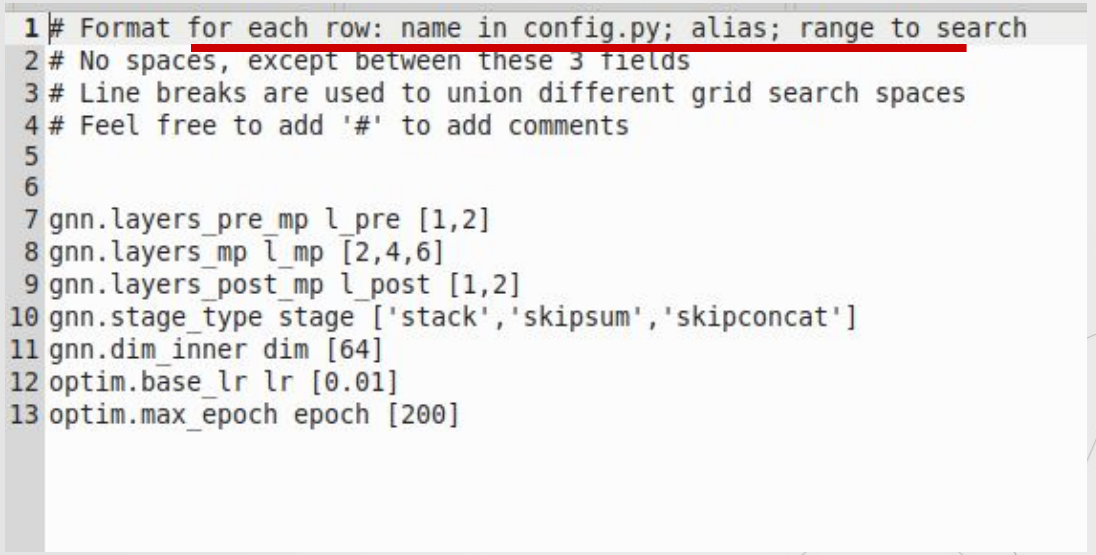
04 Example: batch training

Idea define multiple experiments by perturbing a base config file



```yaml
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

+



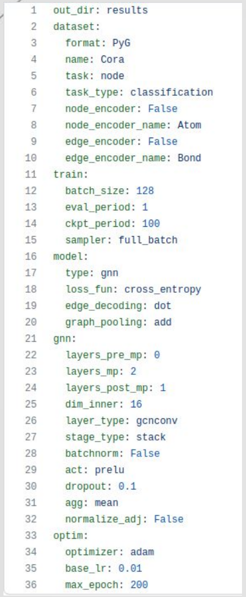
```
# Format for each row: name in config.py; alias; range to search
# No spaces, except between these 3 fields
# Line breaks are used to union different grid search spaces
# Feel free to add '#' to add comments


gnn.layers_pre_mp l_pre [1,2]
gnn.layers_mp l_mp [2,4,6]
gnn.layers_post_mp l_post [1,2]
gnn.stage_type stage ['stack','skipsum','skipconcat']
gnn.dim_inner dim [64]
optim.base_lr lr [0.01]
optim.max_epoch epoch [200]
```

configs/pyg/example_node.yaml

grids/pyg/example.txt

04 Example: batch training

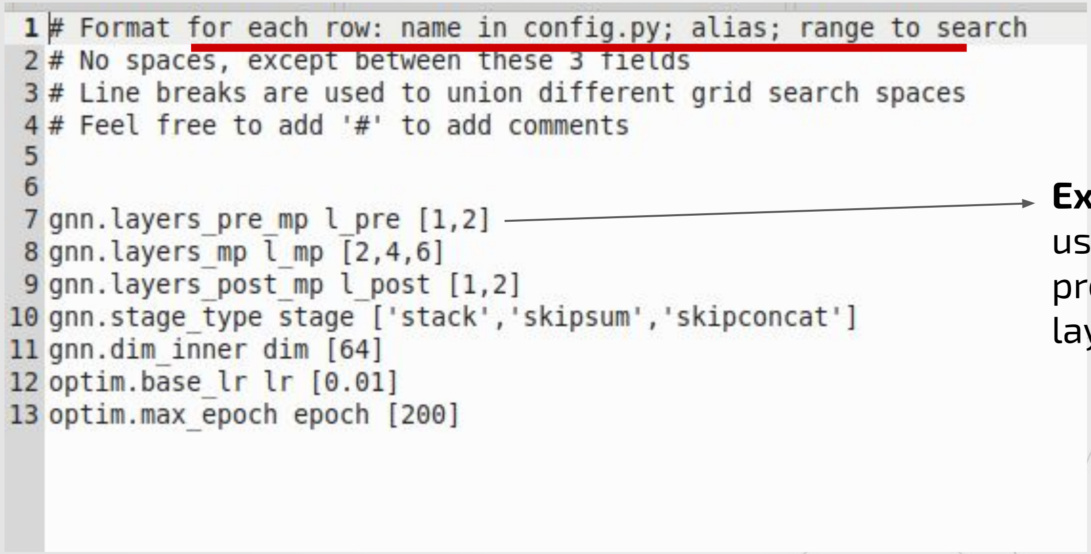
Idea define multiple experiments by perturbing a base config file



```
out_dir: results
dataset:
  format: PyG
  name: Cora
  task: node
  task_type: classification
  node_encoder: False
  node_encoder_name: Atom
  edge_encoder: False
  edge_encoder_name: Bond
train:
  batch_size: 128
  eval_period: 1
  ckpt_period: 100
  sampler: full_batch
model:
  type: gnn
  loss_fun: cross_entropy
  edge_decoding: dot
  graph_pooling: add
gnn:
  layers_pre_mp: 0
  layers_mp: 2
  layers_post_mp: 1
  dim_inner: 16
  layer_type: gcnconv
  stage_type: stack
  batchnorm: False
  act: prelu
  dropout: 0.1
  agg: mean
  normalize_adj: False
optim:
  optimizer: adam
  base_lr: 0.01
  max_epoch: 200
```

configs/pyg/example_node.yaml

+



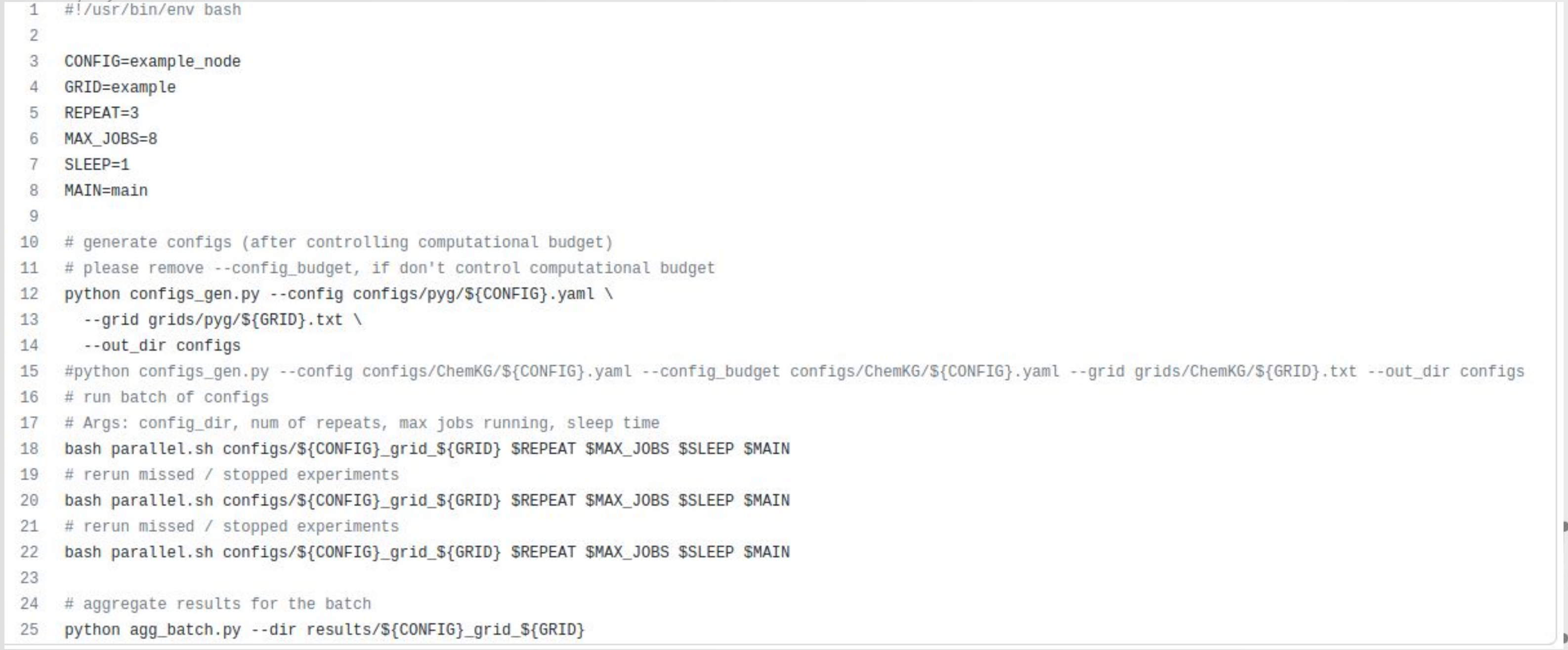
```
# Format for each row: name in config.py; alias; range to search
# No spaces, except between these 3 fields
# Line breaks are used to union different grid search spaces
# Feel free to add '#' to add comments


gnn.layers_pre_mp l_pre [1,2]
gnn.layers_mp l_mp [2,4,6]
gnn.layers_post_mp l_post [1,2]
gnn.stage_type stage ['stack','skipsum','skipconcat']
gnn.dim_inner dim [64]
optim.base_lr lr [0.01]
optim.max_epoch epoch [200]
```

Example:
use 1 or 2 preprocessing layers

grids/pyg/example.txt

04 Example: batch training

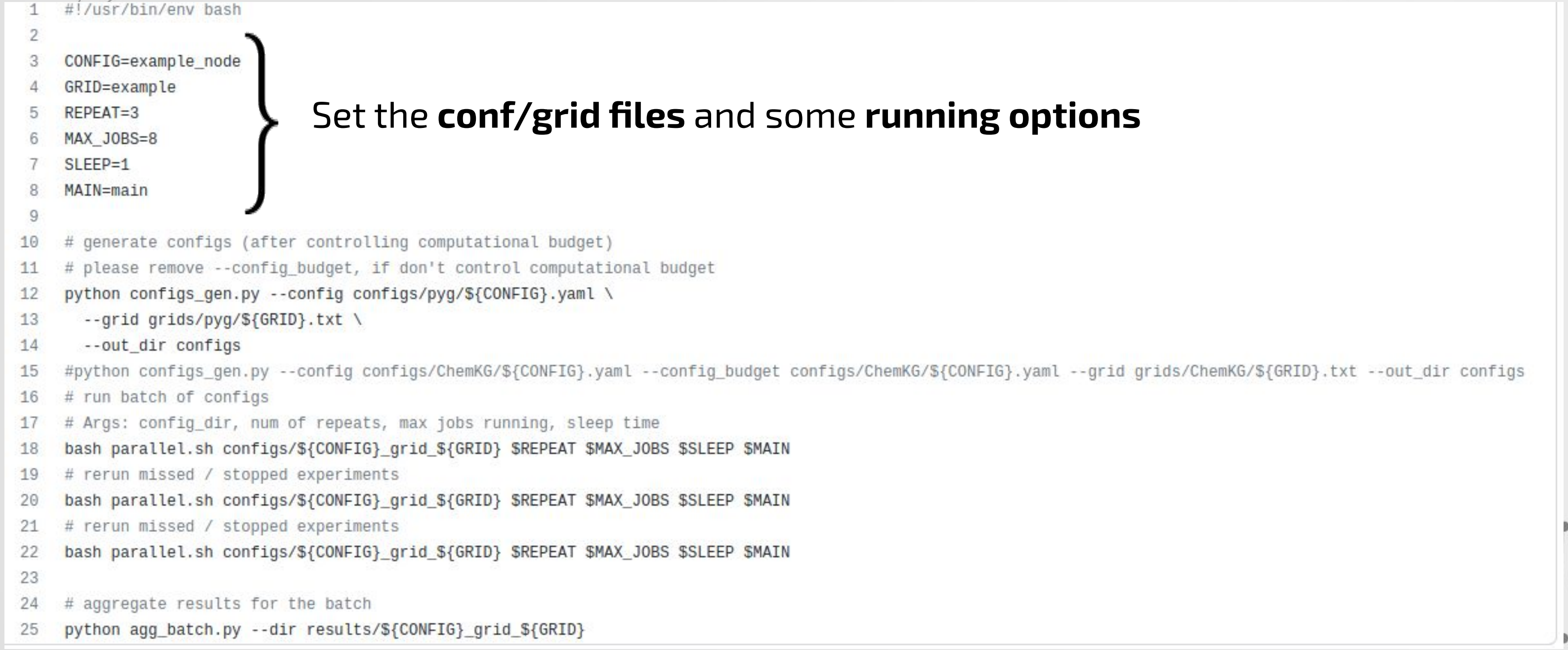
Run run_batch.sh . . .



```bash
#!/usr/bin/env bash

CONFIG=example_node
GRID=example
REPEAT=3
MAX_JOBS=8
SLEEP=1
MAIN=main

# generate configs (after controlling computational budget)
# please remove --config_budget, if don't control computational budget
python configs_gen.py --config configs/pyg/${CONFIG}.yaml \
  --grid grids/pyg/${GRID}.txt \
  --out_dir configs
#python configs_gen.py --config configs/ChemKG/${CONFIG}.yaml --config_budget configs/ChemKG/${CONFIG}.yaml --grid grids/ChemKG/${GRID}.txt --out_dir configs
# run batch of configs
# Args: config_dir, num of repeats, max jobs running, sleep time
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN

# aggregate results for the batch
python agg_batch.py --dir results/${CONFIG}_grid_${GRID}
```

04 Example: batch training

Run run_batch.sh . . .

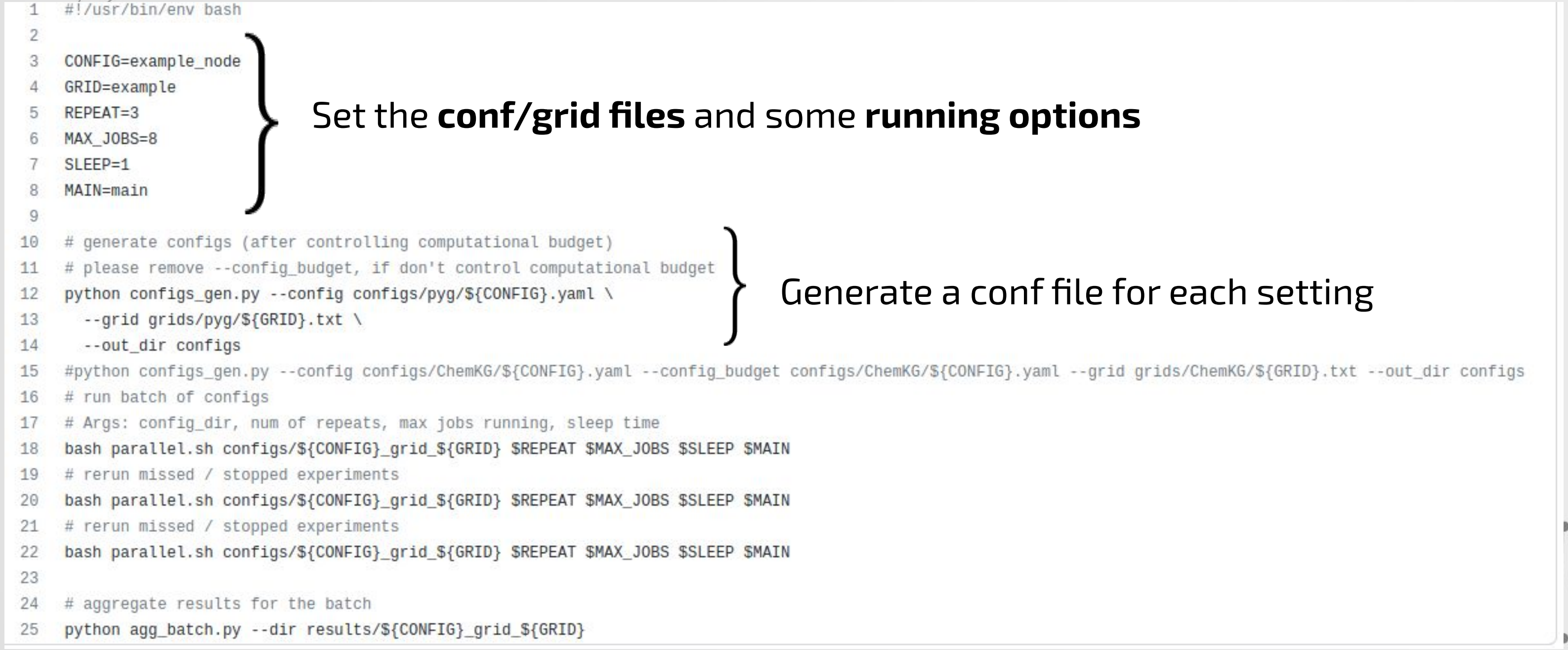


```bash
#!/usr/bin/env bash

CONFIG=example_node
GRID=example
REPEAT=3
MAX_JOBS=8
SLEEP=1
MAIN=main

# generate configs (after controlling computational budget)
# please remove --config_budget, if don't control computational budget
python configs_gen.py --config configs/pyg/${CONFIG}.yaml \
  --grid grids/pyg/${GRID}.txt \
  --out_dir configs
#python configs_gen.py --config configs/ChemKG/${CONFIG}.yaml --config_budget configs/ChemKG/${CONFIG}.yaml --grid grids/ChemKG/${GRID}.txt --out_dir configs
# run batch of configs
# Args: config_dir, num of repeats, max jobs running, sleep time
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN

# aggregate results for the batch
python agg_batch.py --dir results/${CONFIG}_grid_${GRID}
```

Set the **conf/grid files** and some **running options**

04 Example: batch training

Run run_batch.sh . . .



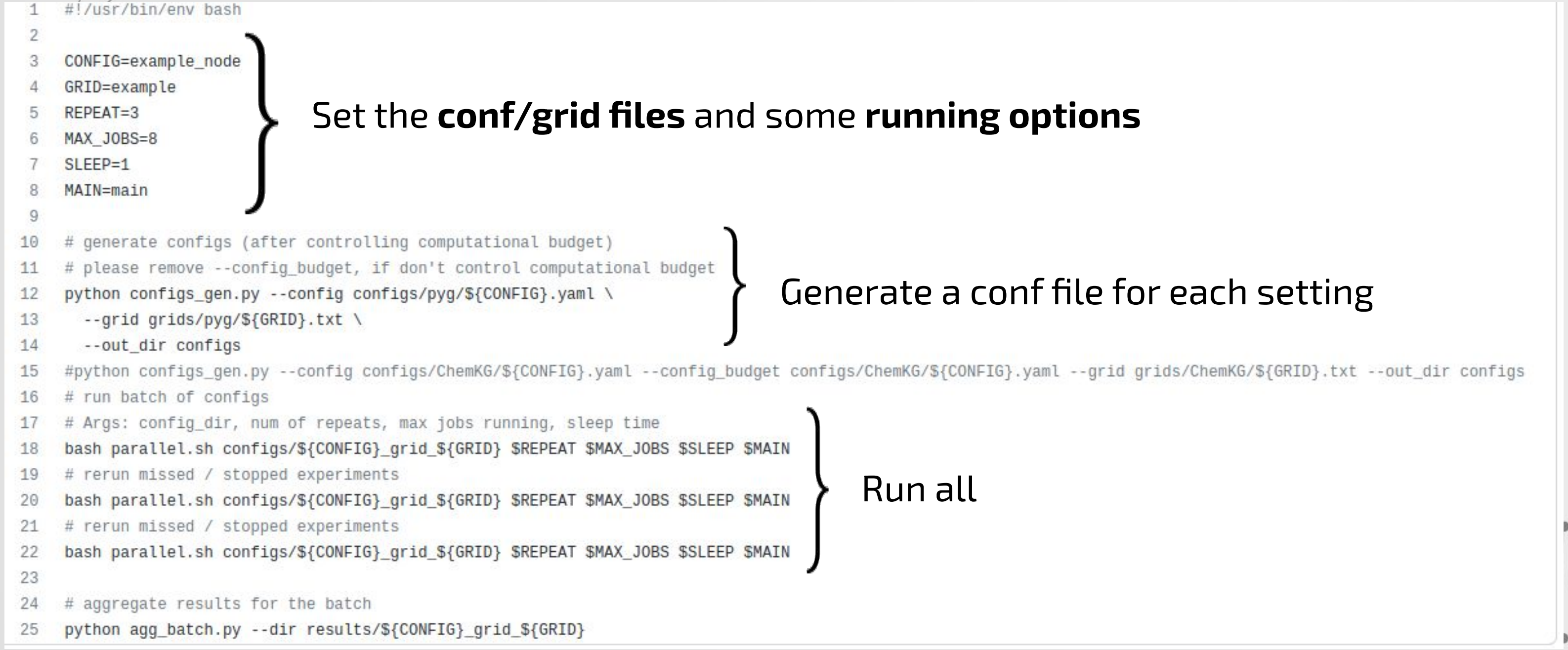
```
#!/usr/bin/env bash

CONFIG=example_node
GRID=example
REPEAT=3
MAX_JOBS=8
SLEEP=1
MAIN=main

# generate configs (after controlling computational budget)
# please remove --config_budget, if don't control computational budget
python configs_gen.py --config configs/pyg/${CONFIG}.yaml \
  --grid grids/pyg/${GRID}.txt \
  --out_dir configs
#python configs_gen.py --config configs/ChemKG/${CONFIG}.yaml --config_budget configs/ChemKG/${CONFIG}.yaml --grid grids/ChemKG/${GRID}.txt --out_dir configs
# run batch of configs
# Args: config_dir, num of repeats, max jobs running, sleep time
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN

# aggregate results for the batch
python agg_batch.py --dir results/${CONFIG}_grid_${GRID}
```

Set the **conf/grid files** and some **running options**

Generate a conf file for each setting

04 Example: batch training

Run run_batch.sh . . .



```
#!/usr/bin/env bash

CONFIG=example_node
GRID=example
REPEAT=3
MAX_JOBS=8
SLEEP=1
MAIN=main

# generate configs (after controlling computational budget)
# please remove --config_budget, if don't control computational budget
python configs_gen.py --config configs/pyg/${CONFIG}.yaml \
  --grid grids/pyg/${GRID}.txt \
  --out_dir configs
#python configs_gen.py --config configs/ChemKG/${CONFIG}.yaml --config_budget configs/ChemKG/${CONFIG}.yaml --grid grids/ChemKG/${GRID}.txt --out_dir configs
# run batch of configs
# Args: config_dir, num of repeats, max jobs running, sleep time
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN

# aggregate results for the batch
python agg_batch.py --dir results/${CONFIG}_grid_${GRID}
```

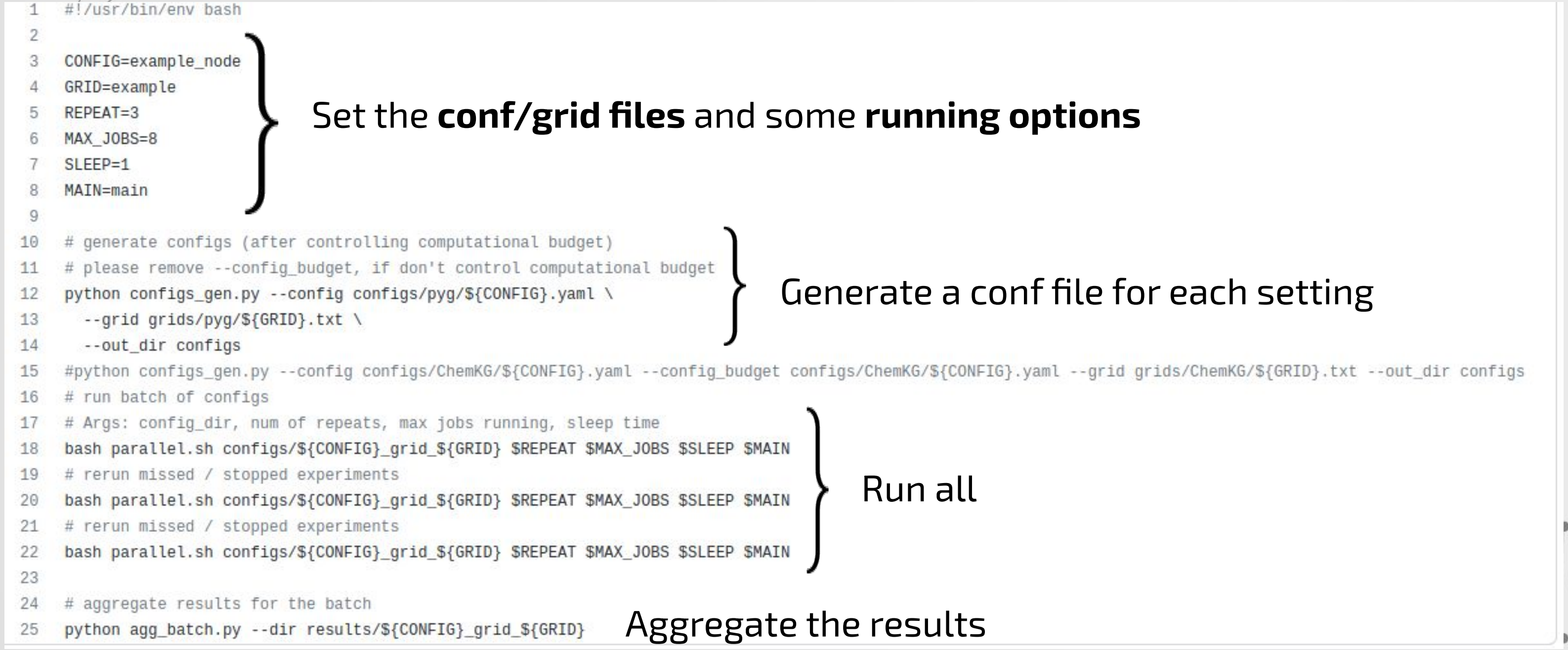
Set the **conf/grid files** and some **running options**

Generate a conf file for each setting

Run all

04 Example: batch training

Run run_batch.sh . . .



```bash
#!/usr/bin/env bash

CONFIG=example_node
GRID=example
REPEAT=3
MAX_JOBS=8
SLEEP=1
MAIN=main

# generate configs (after controlling computational budget)
# please remove --config_budget, if don't control computational budget
python configs_gen.py --config configs/pyg/${CONFIG}.yaml \
  --grid grids/pyg/${GRID}.txt \
  --out_dir configs
#python configs_gen.py --config configs/ChemKG/${CONFIG}.yaml --config_budget configs/ChemKG/${CONFIG}.yaml --grid grids/ChemKG/${GRID}.txt --out_dir configs
# run batch of configs
# Args: config_dir, num of repeats, max jobs running, sleep time
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN
# rerun missed / stopped experiments
bash parallel.sh configs/${CONFIG}_grid_${GRID} $REPEAT $MAX_JOBS $SLEEP $MAIN

# aggregate results for the batch
python agg_batch.py --dir results/${CONFIG}_grid_${GRID}
```

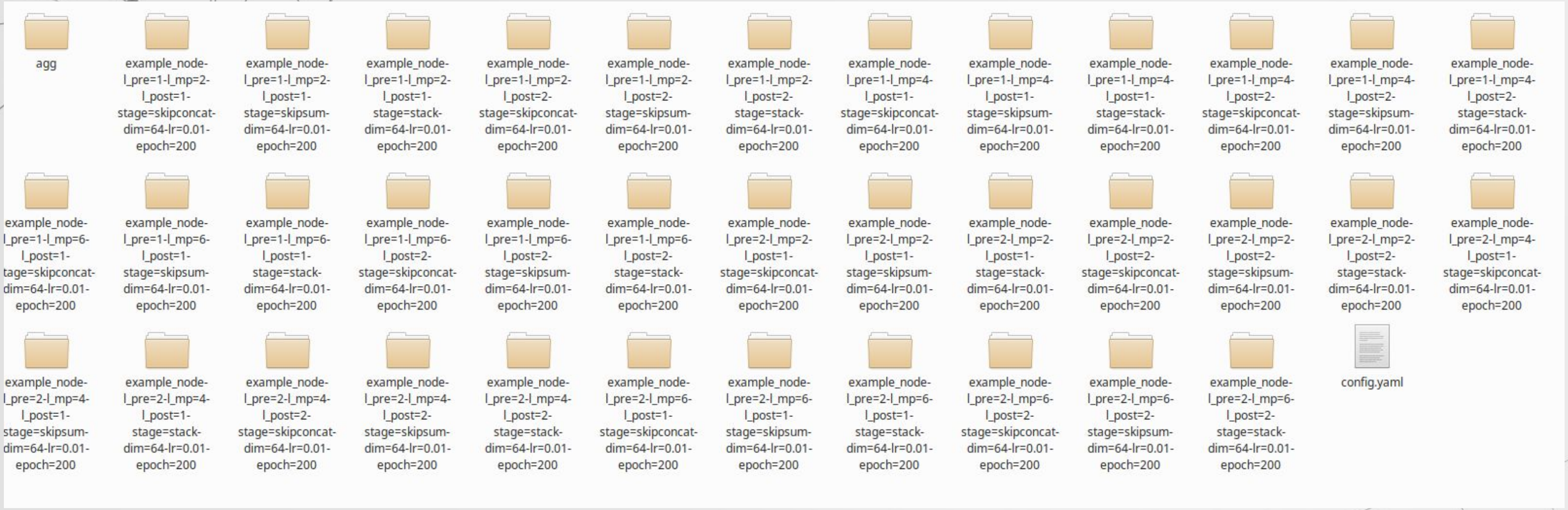
Set the **conf/grid files** and some **running options**

Generate a conf file for each setting

Run all

Aggregate the results

04 Example: batch training



agg

example_node-l_pre=1-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=6-l_post=1-tage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=1-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200

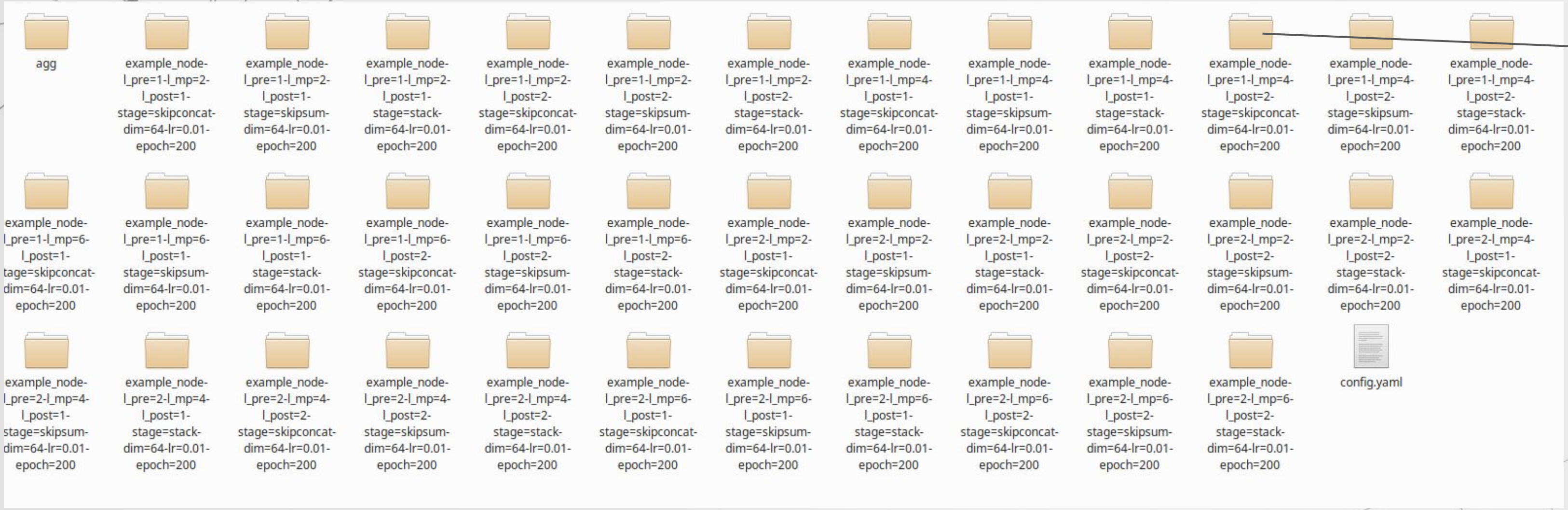
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200

example_node-l_pre=2-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200

config.yaml

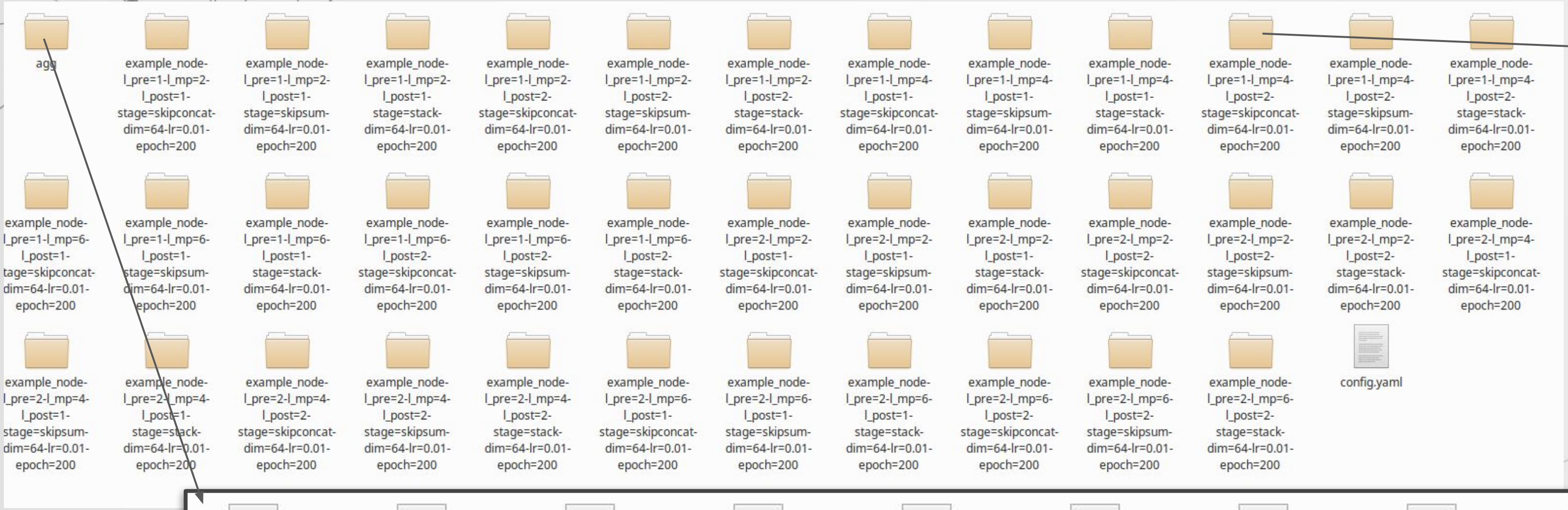
04 Example: batch training



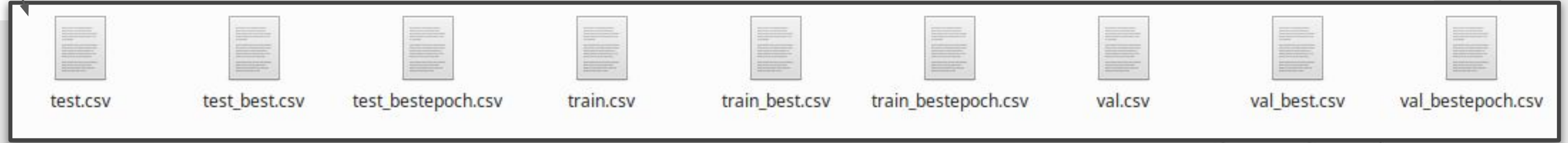
```
agg
example_node-l_pre=1-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-tage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
config.yaml
```

Single experiment: same as before

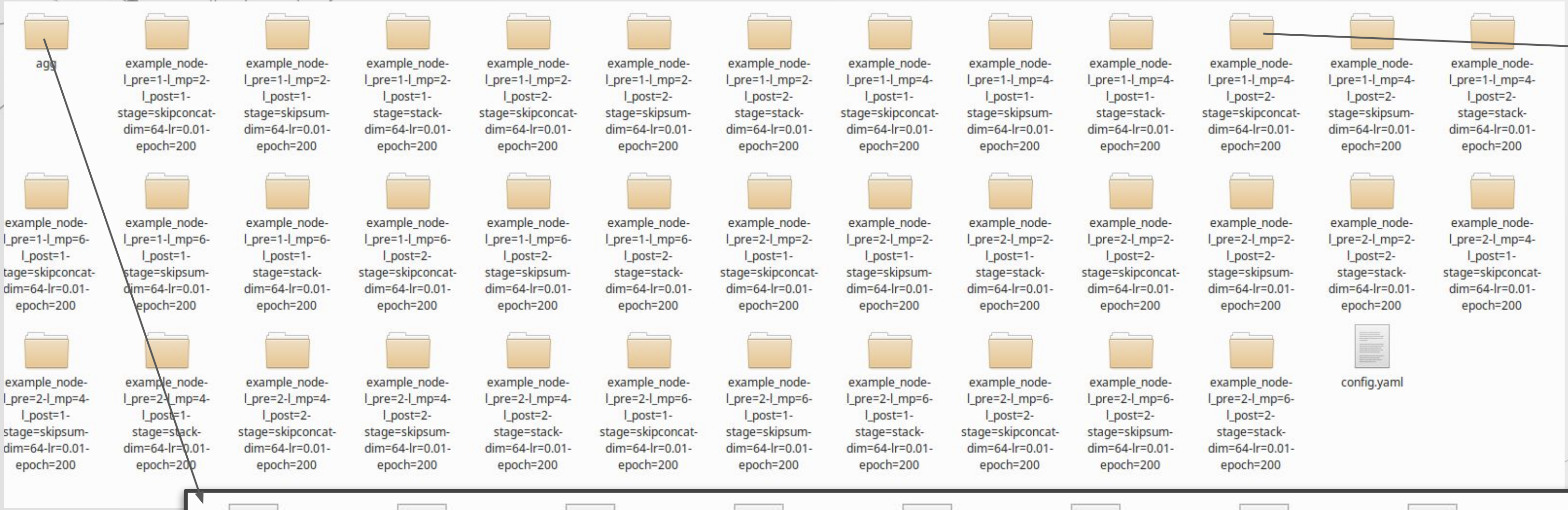
04 Example: batch training


agg
example_node-l_pre=1-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
config.yaml

Single experiment: same as before

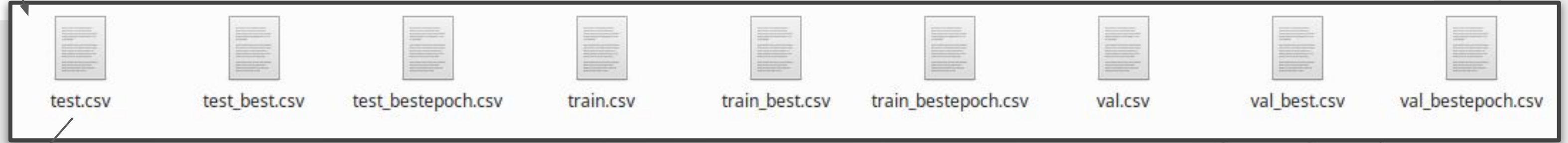

test.csv
test_best.csv
test_bestepoch.csv
train.csv
train_best.csv
train_bestepoch.csv
val.csv
val_best.csv
val_bestepoch.csv

04 Example: batch training



agg
example_node-l_pre=1-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
config.yaml

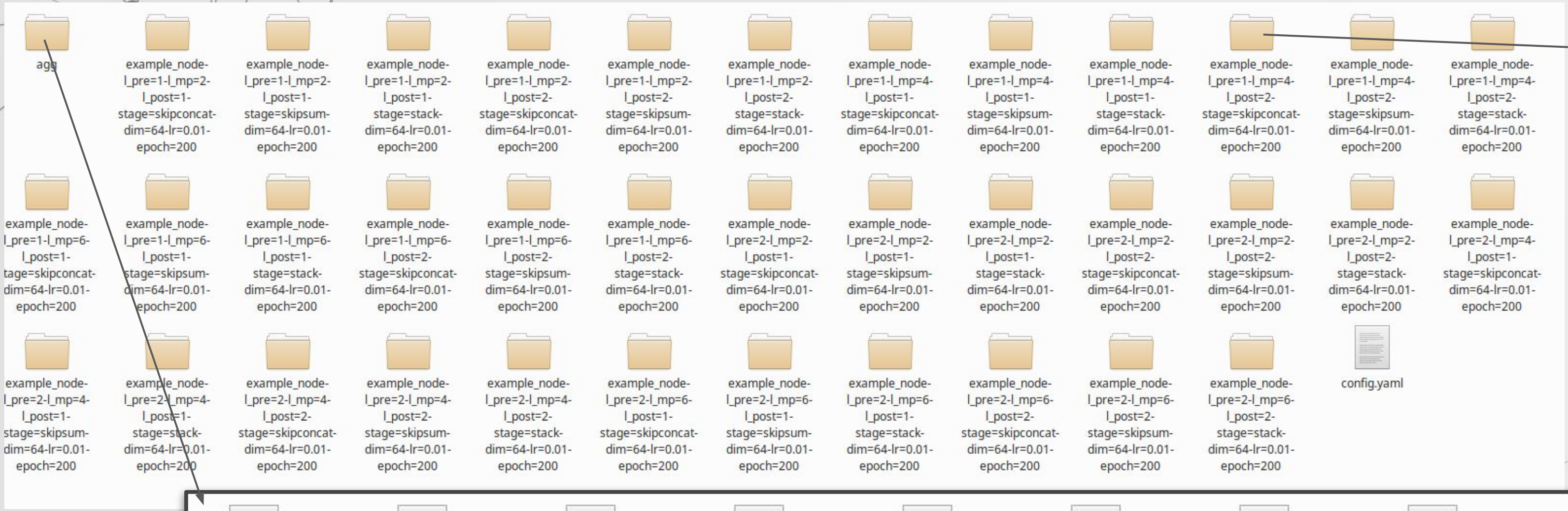
Single experiment: same as before



test.csv
test_best.csv
test_bestepoch.csv
train.csv
train_best.csv
train_bestepoch.csv
val.csv
val_best.csv
val_bestepoch.csv

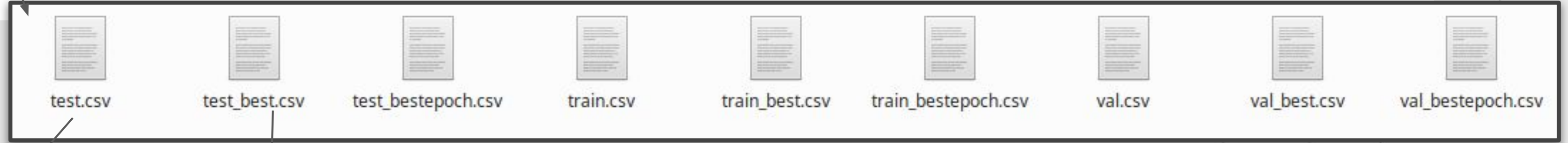
Final epoch

04 Example: batch training



agg
example_node-l_pre=1-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=1-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
example_node-l_pre=2-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
config.yaml

Single experiment: same as before

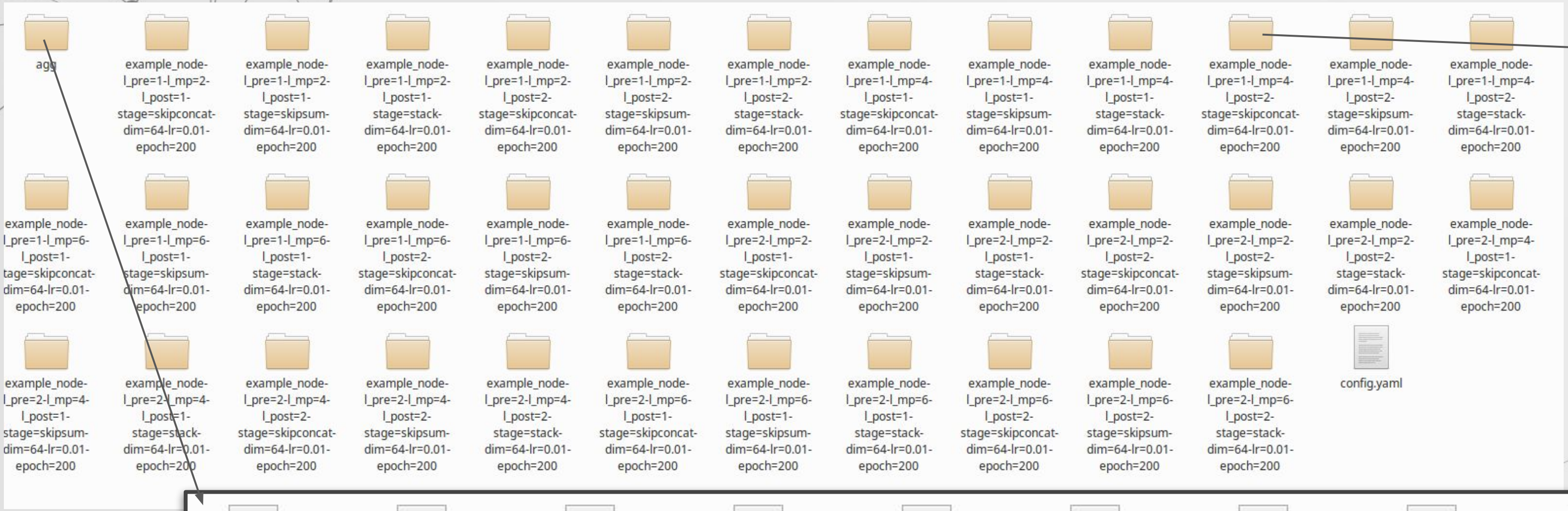


test.csv
test_best.csv
test_bestepoch.csv
train.csv
train_best.csv
train_bestepoch.csv
val.csv
val_best.csv
val_bestepoch.csv

Final epoch

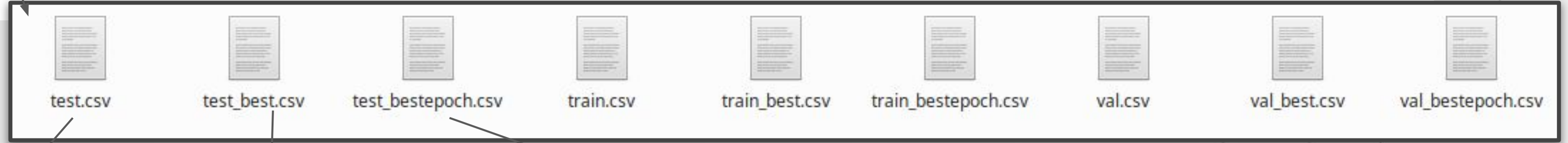
Epoch with the highest average validation accuracy

04 Example: batch training



- agg
- example_node-l_pre=1-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=6-l_post=1-tage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=1-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=2-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=2-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=2-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=2-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=2-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=2-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=4-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=4-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=4-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=4-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=4-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=4-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=6-l_post=1-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=6-l_post=1-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=6-l_post=1-stage=stack-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=6-l_post=2-stage=skipconcat-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=6-l_post=2-stage=skipsum-dim=64-lr=0.01-epoch=200
- example_node-l_pre=2-l_mp=6-l_post=2-stage=stack-dim=64-lr=0.01-epoch=200
- config.yaml

Single experiment: same as before



- test.csv
- test_best.csv
- test_bestepoch.csv
- train.csv
- train_best.csv
- train_bestepoch.csv
- val.csv
- val_best.csv
- val_bestepoch.csv

Final epoch

Epoch with the highest average validation accuracy

Epoch with the highest validation accuracy averaged over different random seeds

04 Example: batch training

Warning

small bug: the script run_batch.sh does not create the folder for the results. You need to create it manually if you want to run another experiment with another name

04 Example: batch training

Warning

small bug: the script run_batch.sh does not create the folder for the results. You need to create it manually if you want to run another experiment with another name

Warning

unexplored feature: there is also an option

```
--config-budget conf.yaml
```

that uses a single conf file / architecture as a budget constraint -> all the other tested models are adjusted to this budget

04 Example: batch training

Warning

small bug: the script run_batch.sh does not create the folder for the results. You need to create it manually if you want to run another experiment with another name

Warning

unexplored feature: there is also an option

```
--config-budget conf.yaml
```

that uses a single conf file / architecture as a budget constraint -> all the other tested models are adjusted to this budget

Still not very well documented, errors when running

04 Example: batch training

Warning

small bug: the script run_batch.sh does not create the folder for the results. You need to create it manually if you want to run another experiment with another name

Warning

unexplored feature: there is also an option

```
--config-budget conf.yaml
```

that uses a single conf file / architecture as a budget constraint -> all the other tested models are adjusted to this budget → Updates in future tutorials!

Still not very well documented, errors when running

05 Customization

Add new modules, losses, options…:

- **Personal use:** add to your local `graphgym/custom_graphgym`

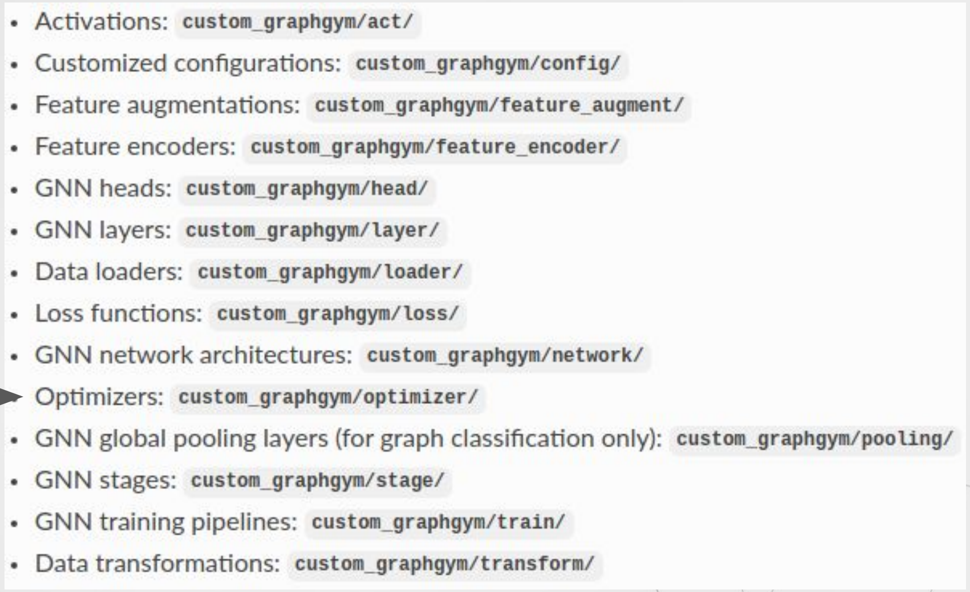
- **Contribution to the project:** add to PyG `torch_geometric/graphgym/contrib` + pull request

05 Customization

Add new modules, losses, options...:

- **Personal use:** add to your local `graphgym/custom_graphgym`

- **Contribution to the project:** add to PyG `torch_geometric/graphgym/contrib` + pull request



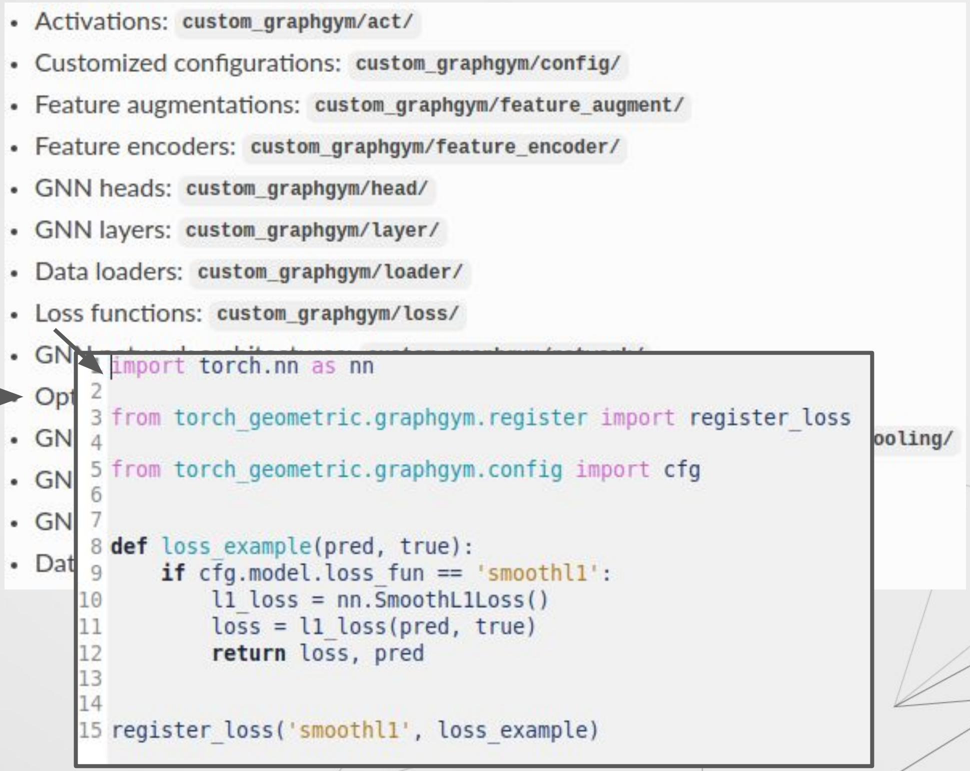
- Activations: `custom_graphgym/act/`
- Customized configurations: `custom_graphgym/config/`
- Feature augmentations: `custom_graphgym/feature_augment/`
- Feature encoders: `custom_graphgym/feature_encoder/`
- GNN heads: `custom_graphgym/head/`
- GNN layers: `custom_graphgym/layer/`
- Data loaders: `custom_graphgym/loader/`
- Loss functions: `custom_graphgym/loss/`
- GNN network architectures: `custom_graphgym/network/`
- Optimizers: `custom_graphgym/optimizer/`
- GNN global pooling layers (for graph classification only): `custom_graphgym/pooling/`
- GNN stages: `custom_graphgym/stage/`
- GNN training pipelines: `custom_graphgym/train/`
- Data transformations: `custom_graphgym/transform/`

05 Customization

Add new modules, losses, options...:

- **Personal use:** add to your local `graphgym/custom_graphgym`

- **Contribution to the project:** add to PyG `torch_geometric/graphgym/contrib` + pull request



- Activations: `custom_graphgym/act/`
- Customized configurations: `custom_graphgym/config/`
- Feature augmentations: `custom_graphgym/feature_augment/`
- Feature encoders: `custom_graphgym/feature_encoder/`
- GNN heads: `custom_graphgym/head/`
- GNN layers: `custom_graphgym/layer/`
- Data loaders: `custom_graphgym/loader/`
- Loss functions: `custom_graphgym/loss/`
- GN
- Opt
- GN `ooling/`
- GN
- GN
- Dat

```python
import torch.nn as nn

from torch_geometric.graphgym.register import register_loss

from torch_geometric.graphgym.config import cfg


def loss_example(pred, true):
    if cfg.model.loss_fun == 'smoothl1':
        l1_loss = nn.SmoothL1Loss()
        loss = l1_loss(pred, true)
        return loss, pred


register_loss('smoothl1', loss_example)
```

graphgym/custom_graphgym/loss/example.py

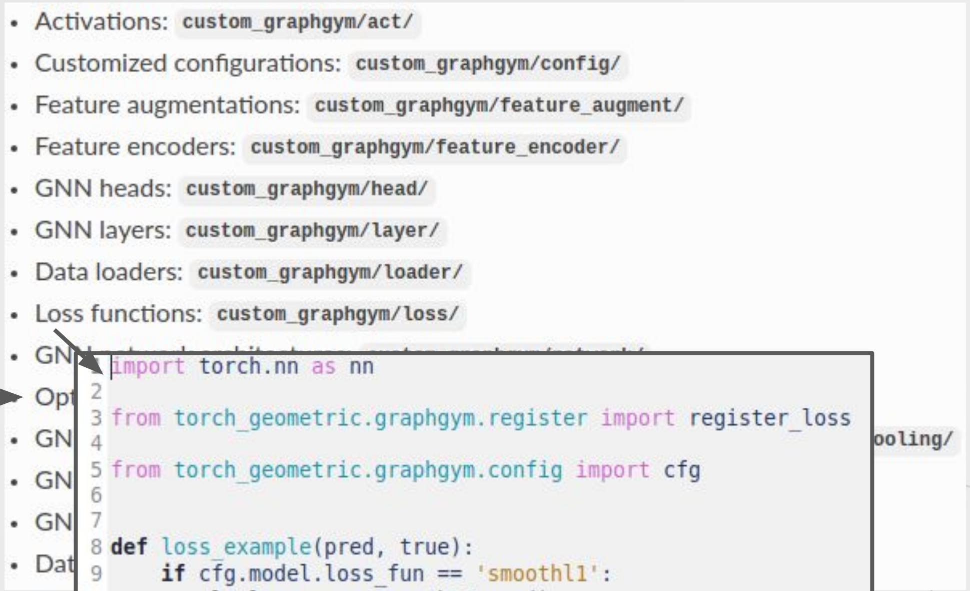
05 Customization

Add new modules, losses, options...:

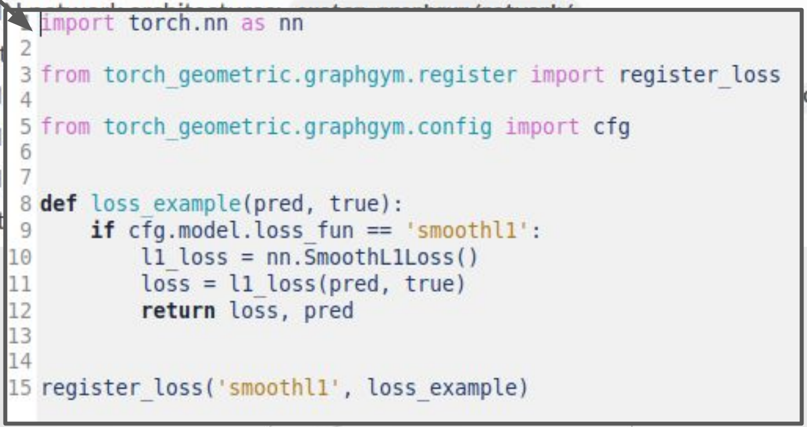
- **Personal use:** add to your local `graphgym/custom_graphgym`

- **Contribution to the project:** add to PyG `torch_geometric/graphgym/contrib` + pull request

(also possible to add new config fields)



- Activations: `custom_graphgym/act/`
- Customized configurations: `custom_graphgym/config/`
- Feature augmentations: `custom_graphgym/feature_augment/`
- Feature encoders: `custom_graphgym/feature_encoder/`
- GNN heads: `custom_graphgym/head/`
- GNN layers: `custom_graphgym/layer/`
- Data loaders: `custom_graphgym/loader/`
- Loss functions: `custom_graphgym/loss/`
- GN
- Opt
- GN `ooling/`
- GN
- GN
- Dat



```python
import torch.nn as nn

from torch_geometric.graphgym.register import register_loss

from torch_geometric.graphgym.config import cfg


def loss_example(pred, true):
    if cfg.model.loss_fun == 'smoothl1':
        l1_loss = nn.SmoothL1Loss()
        loss = l1_loss(pred, true)
        return loss, pred


register_loss('smoothl1', loss_example)
```

graphgym/custom_graphgym/loss/example.py

THANKS

Questions?

gsantin@fbk.eu