

---

# DPGEN2

DeepModeling

Oct 04, 2022



# USER GUIDE

<b>1</b>	<b>Guide on dpgen2 commands</b>	<b>3</b>
1.1	Submit a workflow . . . . .	3
1.2	Check the convergence of a workflow . . . . .	3
1.3	Watch the progress of a workflow . . . . .	3
1.4	Show the keys of steps . . . . .	4
1.5	Resubmit a workflow . . . . .	5
<b>2</b>	<b>Command line interface</b>	<b>7</b>
2.1	Named Arguments . . . . .	7
2.2	Valid subcommands . . . . .	7
2.3	Sub-commands . . . . .	7
2.3.1	submit . . . . .	7
2.3.2	resubmit . . . . .	8
2.3.3	showkey . . . . .	8
2.3.4	status . . . . .	8
2.3.5	download . . . . .	9
2.3.6	watch . . . . .	9
<b>3</b>	<b>Guide on writing input scripts for dpgen2 commands</b>	<b>11</b>
3.1	Preliminaries . . . . .	11
3.2	The input script for all dpgen2 commands . . . . .	11
3.3	The input script for submit and resubmit . . . . .	11
3.3.1	Inputs . . . . .	12
3.3.2	Training . . . . .	12
3.3.3	Exploration . . . . .	12
3.3.4	FP . . . . .	13
3.3.5	Configuration of dflow step . . . . .	14
<b>4</b>	<b>Arguments of the submit script</b>	<b>15</b>
<b>5</b>	<b>OP Configs</b>	<b>51</b>
5.1	RunDPTrain . . . . .	51
5.2	RunLmp . . . . .	52
5.3	RunVasp . . . . .	52
<b>6</b>	<b>Alloy configs</b>	<b>53</b>
<b>7</b>	<b>Step Configs</b>	<b>55</b>
7.1	Configurations for dflow steps . . . . .	55
<b>8</b>	<b>Developers' guide</b>	<b>59</b>

8.1	The concurrent learning algorithm . . . . .	59
8.2	Overview of the DPGEN2 Implementation . . . . .	60
8.3	The DPGEN2 workflow . . . . .	60
8.3.1	Inside the block operator . . . . .	61
8.3.2	The exploration strategy . . . . .	61
8.4	How to contribute . . . . .	62
<b>9</b>	<b>Operators</b>	<b>63</b>
9.1	The super-OP PrepRunDPTrain . . . . .	63
9.2	The OP RunDPTrain . . . . .	66
<b>10</b>	<b>Exploration</b>	<b>69</b>
10.1	Stage scheduler . . . . .	69
10.2	Exploration task groups . . . . .	70
10.3	Configuration selector . . . . .	71
<b>11</b>	<b>DPGEN2 API</b>	<b>73</b>
11.1	dpgen2 package . . . . .	73
11.1.1	Subpackages . . . . .	73
11.1.2	Submodules . . . . .	112
11.1.3	dpgen2.constants module . . . . .	112
	<b>Python Module Index</b>	<b>113</b>
	<b>Index</b>	<b>115</b>

DPGEN2 is the 2nd generation of the Deep Potential GENerator.

---

**Important:** The project DeePMD-kit is licensed under [GNU LGPLv3.0](#).

---



## GUIDE ON DPGEN2 COMMANDS

One may use dpngen2 through command line interface. A full documentation of the cli is found [here](#)

### 1.1 Submit a workflow

The dpngen2 workflow can be submitted via the `submit` command

```
dpngen2 submit input.json
```

where `input.json` is the input script. A guide of writing the script is found [here](#). When a workflow is submitted, a ID (WFID) of the workflow will be printed for later reference.

### 1.2 Check the convergence of a workflow

The convergence of stages of the workflow can be checked by the `status` command. It prints the indexes of the finished stages, iterations, and the accurate, candidate and failed ratio of explored configurations of each iteration.

```
$ dpngen2 status input.json WFID
#  stage  id_stg.  iter.      accu.      cand.      fail.
# Stage   0  -----
#         0       0       0       0.8333      0.1667      0.0000
#         0       1       1       0.7593      0.2407      0.0000
#         0       2       2       0.7778      0.2222      0.0000
#         0       3       3       1.0000      0.0000      0.0000
# Stage   0  converged YES  reached max numb iterations NO
# All stages converged
```

### 1.3 Watch the progress of a workflow

The progress of a workflow can be watched on-the-fly

```
$ dpngen2 watch input.json WFID
INFO:root:steps iter-000000--prep-run-train----- finished
INFO:root:steps iter-000000--prep-run-lmp----- finished
INFO:root:steps iter-000000--prep-run-fp----- finished
INFO:root:steps iter-000000--collect-data----- finished
```

(continues on next page)

(continued from previous page)

```
INFO:root:steps iter-000001--prep-run-train----- finished
INFO:root:steps iter-000001--prep-run-lmp----- finished
...
```

The artifacts can be downloaded on-the-fly with `-d` flag.

## 1.4 Show the keys of steps

Each `dpngen2` step is assigned a unique key. The keys of the finished steps can be checked with `showkey` command

```

0 : init--scheduler
1 : init--id
2 : iter-000000--prep-train
3 -> 6 : iter-000000--run-train-0000 -> iter-000000--run-train-0003
7 : iter-000000--prep-run-train
8 : iter-000000--prep-lmp
9 -> 17 : iter-000000--run-lmp-000000 -> iter-000000--run-lmp-000008
18 : iter-000000--prep-run-lmp
19 : iter-000000--select-confs
20 : iter-000000--prep-fp
21 -> 24 : iter-000000--run-fp-000000 -> iter-000000--run-fp-000003
25 : iter-000000--prep-run-fp
26 : iter-000000--collect-data
27 : iter-000000--block
28 : iter-000000--scheduler
29 : iter-000000--id
30 : iter-000001--prep-train
31 -> 34 : iter-000001--run-train-0000 -> iter-000001--run-train-0003
35 : iter-000001--prep-run-train
36 : iter-000001--prep-lmp
37 -> 45 : iter-000001--run-lmp-000000 -> iter-000001--run-lmp-000008
46 : iter-000001--prep-run-lmp
47 : iter-000001--select-confs
48 : iter-000001--prep-fp
49 -> 52 : iter-000001--run-fp-000000 -> iter-000001--run-fp-000003
53 : iter-000001--prep-run-fp
54 : iter-000001--collect-data
55 : iter-000001--block
56 : iter-000001--scheduler
57 : iter-000001--id
```



## 1.5 Resubmit a workflow

If a workflow stopped abnormally, one may submit a new workflow with some steps of the old workflow reused.

```
dpgen2 resubmit input.json WFID --reuse 0-49
```

The steps of workflow WDID 0-49 will be reused in the new workflow. The indexes of the steps are printed by `dpgen2 showkey`. In the example, all the steps before the `iter-000001--run-fp-000000` will be used in the new workflow.



## COMMAND LINE INTERFACE

DPGEN2: concurrent learning workflow generating the machine learning potential energy models.

```
usage: dpgen2 [-h] [--version]
             {submit,resubmit,showkey,status,download,watch} ...
```

### 2.1 Named Arguments

**--version**            show program's version number and exit

### 2.2 Valid subcommands

**command**            Possible choices: submit, resubmit, showkey, status, download, watch

### 2.3 Sub-commands

#### 2.3.1 submit

Submit DPGEN2 workflow

```
dpgen2 submit [-h] [-o] CONFIG
```

#### Positional Arguments

**CONFIG**            the config file in json format defining the workflow.

### Named Arguments

**-o, --old-compatible** compatible with old-style input script used in dpngen2 < 0.0.6.  
Default: False

### 2.3.2 resubmit

Submit DPGEN2 workflow resuing steps from an existing workflow

```
dpngen2 resubmit [-h] [-l] [--reuse REUSE [REUSE ...]] [-o] CONFIG ID
```

### Positional Arguments

**CONFIG** the config file in json format defining the workflow.  
**ID** the ID of the existing workflow.

### Named Arguments

**-l, --list** list the Steps of the existing workflow.  
Default: False  
**--reuse** specify which Steps to reuse.  
**-o, --old-compatible** compatible with old-style input script used in dpngen2 < 0.0.6.  
Default: False

### 2.3.3 showkey

Print the keys of the successful DPGEN2 steps

```
dpngen2 showkey [-h] CONFIG ID
```

### Positional Arguments

**CONFIG** the config file in json format.  
**ID** the ID of the existing workflow.

### 2.3.4 status

Print the status (stage, iteration, convergence) of the DPGEN2 workflow

```
dpngen2 status [-h] CONFIG ID
```

### Positional Arguments

<b>CONFIG</b>	the config file in json format.
<b>ID</b>	the ID of the existing workflow.

### 2.3.5 download

Download the artifacts of DPGEN2 steps

```
dpngen2 download [-h] [-k KEYS [KEYS ...]] [-p PREFIX] CONFIG ID
```

### Positional Arguments

<b>CONFIG</b>	the config file in json format.
<b>ID</b>	the ID of the existing workflow.

### Named Arguments

<b>-k, --keys</b>	the keys of the downloaded steps. If not provided download all artifacts
<b>-p, --prefix</b>	the prefix of the path storing the download artifacts

### 2.3.6 watch

Watch a DPGEN2 workflow

```
dpngen2 watch [-h] [-k KEYS [KEYS ...]] [-f FREQUENCY] [-d] [-p PREFIX]
              CONFIG ID
```

### Positional Arguments

<b>CONFIG</b>	the config file in json format.
<b>ID</b>	the ID of the existing workflow.

### Named Arguments

<b>-k, --keys</b>	the subkey to watch. For example, 'prep-run-train' 'prep-run-lmp' Default: ['prep-run-train', 'prep-run-lmp', 'prep-run-fp', 'collect-data']
<b>-f, --frequency</b>	the frequency of workflow status query. In unit of second Default: 600.0
<b>-d, --download</b>	whether to download artifacts of a step when it finishes Default: False
<b>-p, --prefix</b>	the prefix of the path storing the download artifacts



## GUIDE ON WRITING INPUT SCRIPTS FOR DPGEN2 COMMANDS

### 3.1 Preliminaries

The reader of this doc is assumed to be familiar with the concurrent learning algorithm that the dpngen2 implements. If not, one may check [this paper](#).

### 3.2 The input script for all dpngen2 commands

For all the dpngen2 commands, one need to provide dflow2 global configurations. For example,

```
"dflow_config" : {  
  "host" : "http://address.of.the.host:port"  
},  
"dflow_s3_config" : {  
  "s3_endpoint" : "address.of.the.s3.sever:port"  
},
```

The dpngen simply pass all keys of "dflow\_config" to dflow.config and all keys of "dflow\_s3\_config" to dflow.s3\_config.

### 3.3 The input script for submit and resubmit

The full documentation of the submit and resubmit script can be [found here](#). This documentation provides a fast guide on how to write the input script.

In the input script of dpngen2 submit and dpngen2 resubmit, one needs to provide the definition of the workflow and how they are executed in the input script. One may find an example input script in the dpngen2 Al-Mg alloy example.

The definition of the workflow can be provided by the following sections:

### 3.3.1 Inputs

This section provides the inputs to start a dpngen2 workflow. An example for the Al-Mg alloy

```
"inputs": {
  "type_map":          ["Al", "Mg"],
  "mass_map":          [27, 24],
  "init_data_sys":     [
    "path/to/init/data/system/0",
    "path/to/init/data/system/1"
  ],
}
```

The key "init\_data\_sys" provides the initial training data to kick-off the training of deep potential (DP) models.

### 3.3.2 Training

This section defines how a model is trained.

```
"train" : {
  "type" : "dp",
  "numb_models" : 4,
  "config" : {},
  "template_script" : {
    "_comment" : "omitted content of template script"
  },
  "_comment" : "all"
}
```

The "type" : "dp" tell the training method is "dp", i.e. calling [DeePMD-kit](#) to train DP models. The "config" key defines the training configs, see [the full documentation](#). The "template\_script" provides the template training script in json format.

### 3.3.3 Exploration

This section defines how the configuration space is explored.

```
"explore" : {
  "type" : "lmp",
  "config" : {
    "command": "lmp -var restart 0"
  },
  "max_numb_iter" : 5,
  "conv_accuracy" : 0.9,
  "fatal_at_max" : false,
  "f_trust_lo": 0.05,
  "f_trust_hi": 0.50,
  "configurations": [
    {
      "lattice" : ["fcc", 4.57],
      "replicate" : [2, 2, 2],
      "numb_confs" : 30,
    }
  ]
}
```

(continues on next page)



(continued from previous page)

```

        "concentration" : [[1.0, 0.0], [0.5, 0.5], [0.0, 1.0]]
    }
    {
        "lattice" : ["fcc", 4.57],
        "replicate" : [3, 3, 3],
        "numb_confs" : 30,
        "concentration" : [[1.0, 0.0], [0.5, 0.5], [0.0, 1.0]]
    }
],
"stages": [
    { "_idx": 0, "ensemble": "nvt", "nsteps": 20, "press": null, "conf_idx": ↵
↵ [0], "temps": [50,100], "trj_freq": 10, "n_sample" : 3 },
    { "_idx": 1, "ensemble": "nvt", "nsteps": 20, "press": null, "conf_idx": ↵
↵ [1], "temps": [50,100], "trj_freq": 10, "n_sample" : 3 }
],
}

```

The "type" : "lmp" means that configurations are explored by LAMMPS DPMD runs. The "config" key defines the lmp configs, see [the full documentation](#). The "configurations" provides the initial configurations (coordinates of atoms and the simulation cell) of the DPMD simulations. It is a list. The elements of the list can be

- list[str]: The strings provides the path to the configuration files.
- dict: Automatic alloy configuration generator. See [the detailed doc](#) of the allowed keys.

The "stages" defines the exploration stages. It is a list of dicts, with each dict defining a stage. The "ensemble", "nsteps", "press", "temps", "traj\_freq" keys are self-explanatory. "conf\_idx" picks initial configurations of DPMD simulations from the "configurations" list, it provides the index of the element in the "configurations" list. "n\_sample" tells the number of configurations randomly sampled from the set picked by "conf\_idx" for each thermodynamic state. All configurations picked by "conf\_idx" has the same possibility to be sampled. The default value of "n\_sample" is null, in this case all picked configurations are sampled. In the example, each stage have 3 samples and 2 thermodynamic states (NVT, T=50 and 100K), then each iteration run 3x2=6 NVT DPMD simulatins.

### 3.3.4 FP

This section defines the first-principle (FP) calculation .

```

"fp" : {
    "type" : "vasp",
    "config" : {
        "command": "source /opt/intel/oneapi/setvars.sh && mpirun -n 16 vasp_std"
    },
    "task_max": 2,
    "pp_files": {"Al" : "vasp/POTCAR.Al", "Mg" : "vasp/POTCAR.Mg"},
    "incar": "vasp/INCAR",
    "_comment" : "all"
}

```

The "type" : "vasp" means that first-principles are VASP calculations. The "config" key defines the vasp configs, see [the full documentation](#). The "task\_max" key defines the maximal number of vasp calculations in each dpgen2 iteration. The "pp\_files" and "incar" keys provides the pseudopotential files and the template incar file.

### 3.3.5 Configuration of dflow step

The execution units of the dpgen2 are the dflow Steps. How each step is executed is defined by the "step\_configs".

```
"step_configs":{
  "prep_train_config" : {
    "_comment" : "content omitted"
  },
  "run_train_config" : {
    "_comment" : "content omitted"
  },
  "prep_explore_config" : {
    "_comment" : "content omitted"
  },
  "run_explore_config" : {
    "_comment" : "content omitted"
  },
  "prep_fp_config" : {
    "_comment" : "content omitted"
  },
  "run_fp_config" : {
    "_comment" : "content omitted"
  },
  "select_confs_config" : {
    "_comment" : "content omitted"
  },
  "collect_data_config" : {
    "_comment" : "content omitted"
  },
  "cl_step_config" : {
    "_comment" : "content omitted"
  },
  "_comment" : "all"
},
```

The configs for prepare training, run training, prepare exploration, run exploration, prepare fp, run fp, select configurations, collect data and concurrent learning steps are given correspondingly.

The readers are referred to *this page* for a full documentation of the step configs.

Any of the config in the step\_configs can be ommitted. If so, the configs of the step is set to the default step configs, which is provided by the following section, for example,

```
"default_step_config" : {
  "template_config" : {
    "image" : "dpgen2:x.x.x"
  }
},
```

The way of writing the default\_step\_config is the same as any step config in the step\_configs. One may refer to *this page* for full documentation.

## ARGUMENTS OF THE SUBMIT SCRIPT

### **fp:**

type: dict

argument path: fp

The configuration for FP

Depending on the value of *type*, different sub args are accepted.

#### **type:**

type: str (flag key)

argument path: fp/type

possible choices: *vasp*

the type of the fp

When *type* is set to *vasp*:

#### **config:**

type: dict, optional, default: {'command': 'vasp', 'log': 'vasp.log',  
'out': 'data'}

argument path: fp[vasp]/config

Configuration of vasp runs

#### **command:**

type: str, optional, default: vasp

argument path: fp[vasp]/config/command

The command of VASP

#### **log:**

type: str, optional, default: vasp.log

argument path: fp[vasp]/config/log

The log file name of VASP

#### **out:**

type: str, optional, default: data

argument path: fp[vasp]/config/out

The output dir name of labeled data. In *deepmd/npz* format provided by *dpdata*.

### **task\_max:**

type: int, optional, default: 10

argument path: `fp[vasp]/task_max`

Maximum number of vasp tasks for each iteration

**pp\_files:**

type: dict

argument path: `fp[vasp]/pp_files`

The pseudopotential files set by a dict, e.g. `{"Al" : "path/to/the/al/pp/file", "Mg" : "path/to/the/mg/pp/file"}`

**incar:**

type: str

argument path: `fp[vasp]/incar`

The pseudopotential files set by a dict, e.g. `{"Al" : "path/to/the/al/pp/file", "Mg" : "path/to/the/mg/pp/file"}`

**explore:**

type: dict

argument path: `explore`

The configuration for exploration

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)

argument path: `explore/type`

possible choices: *lmp*

the type of the exploration

When *type* is set to *lmp*:

**config:**

type: dict, optional, default: `{'command': 'lmp'}`

argument path: `explore[lmp]/config`

Configuration of lmp exploration

**command:**

type: str, optional, default: *lmp*

argument path: `explore[lmp]/config/command`

The command of LAMMPS

**max\_numb\_iter:**

type: int, optional, default: 10

argument path: `explore[lmp]/max_numb_iter`

Maximum number of iterations per stage

**conv\_accuracy:**

type: float, optional, default: 0.9

argument path: `explore[lmp]/conv_accuracy`

Convergence accuracy

**fatal\_at\_max:**

type: `bool`, optional, default: `True`

argument path: `explore[lmp]/fatal_at_max`

Fatal when the number of iteration per stage reaches the *max\_numb\_iter*

**f\_trust\_lo:**

type: `float`

argument path: `explore[lmp]/f_trust_lo`

Lower trust level of force model deviation

**f\_trust\_hi:**

type: `float`

argument path: `explore[lmp]/f_trust_hi`

Higher trust level of force model deviation

**v\_trust\_lo:**

type: `NoneType` | `float`, optional, default: `None`

argument path: `explore[lmp]/v_trust_lo`

Lower trust level of virial model deviation

**v\_trust\_hi:**

type: `NoneType` | `float`, optional, default: `None`

argument path: `explore[lmp]/v_trust_hi`

Higher trust level of virial model deviation

**configuration\_prefix:**

type: `NoneType` | `str`, optional, default: `None`

argument path: `explore[lmp]/configuration_prefix`

The path prefix of lmp initial configurations

**configurations:**

type: `list`, alias: *configuration*

argument path: `explore[lmp]/configurations`

A list of initial configurations.

**stages:**

type: `list`

argument path: `explore[lmp]/stages`

A list of exploration stages.

**train:**

type: dict

argument path: train

The configuration for training

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)

argument path: train/type

possible choices: *dp*

the type of the training

When *type* is set to *dp*:

**config:**

type: dict, optional, default: {'init\_model\_policy': 'no',  
'init\_model\_old\_ratio': 0.9, 'init\_model\_num\_steps': 400000,  
'init\_model\_start\_lr': 0.0001, 'init\_model\_start\_pref\_e':  
0.1, 'init\_model\_start\_pref\_f': 100,  
'init\_model\_start\_pref\_v': 0.0}

argument path: train[dp]/config

Number of models trained for evaluating the model deviation

**init\_model\_policy:**

type: str, optional, default: no

argument path: train[dp]/config/init\_model\_policy

The policy of init-model training. It can be

- 'no': No init-model training. Training from scratch.
- 'yes': Do init-model training.
- 'old\_data\_larger\_than:XXX': Do init-model if the training data size of the previous model is larger than XXX. XXX is an int number.

**init\_model\_old\_ratio:**

type: float, optional, default: 0.9

argument path: train[dp]/config/init\_model\_old\_ratio

The frequency ratio of old data over new data

**init\_model\_num\_steps:**

type: int, optional, default: 400000, alias: *init\_model\_stop\_batch*

argument path: train[dp]/config/init\_model\_num\_steps

The number of training steps when init-model

**init\_model\_start\_lr:**

type: float, optional, default: 0.0001

argument path: train[dp]/config/init\_model\_start\_lr

The start learning rate when init-model

**init\_model\_start\_pref\_e:**

type: float, optional, default: 0.1

argument path: train[dp]/config/init\_model\_start\_pref\_e

The start energy prefactor in loss when init-model

**init\_model\_start\_pref\_f:**

type: int | float, optional, default: 100

argument path: train[dp]/config/init\_model\_start\_pref\_f

The start force prefactor in loss when init-model

**init\_model\_start\_pref\_v:**

type: float, optional, default: 0.0

argument path: train[dp]/config/init\_model\_start\_pref\_v

The start virial prefactor in loss when init-model

**numb\_models:**

type: int, optional, default: 4

argument path: train[dp]/numb\_models

Number of models trained for evaluating the model deviation

**template\_script:**

type: list | dict

argument path: train[dp]/template\_script

Template training script. It can be a *List[Dict]*, the length of which is the same as *numb\_models*. Each template script in the list is used to train a model. Can be a *Dict*, the models share the same template training script.

**inputs:**

type: dict

argument path: inputs

The input parameter and artifacts for dpngen2

**type\_map:**

type: list

argument path: inputs/type\_map

The type map. e.g. ["Al", "Mg"]. Al and Mg will have type 0 and 1, respectively.

**mass\_map:**

type: list

argument path: inputs/mass\_map

The mass map. e.g. [27., 24.]. Al and Mg will be set with mass 27. and 24. amu, respectively.

**init\_data\_prefix:**

type: NoneType | str, optional, default: None

argument path: inputs/init\_data\_prefix

The prefix of initial data systems

**init\_data\_sys:**

type: list  
argument path: inputs/init\_data\_sys  
The prefix of initial data systems

**upload\_python\_package:**

type: NoneType | str, optional, default: None  
argument path: upload\_python\_package  
Upload python package, for debug purpose

**step\_configs:**

type: dict, optional, default: {}  
argument path: step\_configs  
Configurations for executing dflow steps

**prep\_train\_config:**

type: dict, optional, default: {'template\_config': {'image':  
'dptechnology/dpgen2:latest', 'timeout': None,  
'retry\_on\_transient\_error': None,  
'timeout\_as\_transient\_error': False, 'envs': None},  
'continue\_on\_failed': False, 'continue\_on\_num\_success': None,  
'continue\_on\_success\_ratio': None, 'parallelism': None,  
'executor': None}  
argument path: step\_configs/prep\_train\_config  
Configuration for prepare train

**template\_config:**

type: dict, optional, default: {'image':  
'dptechnology/dpgen2:latest'}  
argument path:  
step\_configs/prep\_train\_config/template\_config  
The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default: dptechnology/dpgen2:latest  
argument path: step\_configs/prep\_train\_config/  
template\_config/image  
The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None  
argument path: step\_configs/prep\_train\_config/  
template\_config/timeout  
The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default: None



argument path: `step_configs/prep_train_config/template_config/retry_on_transient_error`

Retry the step if a `TransientError` is raised.

**timeout\_as\_transient\_error:**

type: `bool`, optional, default: `False`

argument path: `step_configs/prep_train_config/template_config/timeout_as_transient_error`

Treat the timeout as `TransientError`.

**envs:**

type: `dict | NoneType`, optional, default: `None`

argument path: `step_configs/prep_train_config/template_config/envs`

The environmental variables.

**continue\_on\_failed:**

type: `bool`, optional, default: `False`

argument path:

`step_configs/prep_train_config/continue_on_failed`

If continue the the step is failed (`FatalError`, `TransientError`, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: `int | NoneType`, optional, default: `None`

argument path: `step_configs/prep_train_config/continue_on_num_success`

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: `NoneType | float`, optional, default: `None`

argument path: `step_configs/prep_train_config/continue_on_success_ratio`

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: `int | NoneType`, optional, default: `None`

argument path:

`step_configs/prep_train_config/parallelism`

The parallelism for the step

**executor:**

type: `dict | NoneType`, optional, default: `None`

argument path: `step_configs/prep_train_config/executor`

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)  
argument path:  
step\_configs/prep\_train\_config/executor/type  
possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to [lebesgue\\_v2](#):

**extra:**

type: dict, optional  
argument path:  
step\_configs/prep\_train\_config/  
executor[lebesgue\_v2]/extra

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional  
argument path:  
step\_configs/prep\_train\_config/  
executor[lebesgue\_v2]/extra/scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
argument path:  
step\_configs/prep\_train\_config/  
executor[lebesgue\_v2]/extra/program\_id

The ID of the program.

**job\_type:**

type: str, optional, default: container  
argument path:  
step\_configs/prep\_train\_config/  
executor[lebesgue\_v2]/extra/job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
argument path:  
step\_configs/prep\_train\_config/  
executor[lebesgue\_v2]/extra/  
template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in Lebesgue.

**run\_train\_config:**

type: dict, optional, default: {'template\_config': {'image':  
'dptechnology/dpgen2:latest', 'timeout': None,  
'retry\_on\_transient\_error': None,  
'timeout\_as\_transient\_error': False, 'envs': None},  
'continue\_on\_failed': False, 'continue\_on\_num\_success':

None, 'continue\_on\_success\_ratio': None, 'parallelism':  
None, 'executor': None}

argument path: step\_configs/run\_train\_config

Configuration for run train

**template\_config:**

type: dict, optional, default: {'image':  
'dptechnology/dpgen2:latest'}

argument path:

step\_configs/run\_train\_config/template\_config

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default:

dptechnology/dpgen2:latest

argument path:

step\_configs/run\_train\_config/  
template\_config/image

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None

argument path:

step\_configs/run\_train\_config/  
template\_config/timeout

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default:  
None

argument path: step\_configs/  
run\_train\_config/template\_config/  
retry\_on\_transient\_error

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False

argument path: step\_configs/  
run\_train\_config/template\_config/  
timeout\_as\_transient\_error

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default:  
None

argument path:

step\_configs/run\_train\_config/  
template\_config/envs

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False  
argument path: `step_configs/run_train_config/continue_on_failed`

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: int | NoneType, optional, default: None  
argument path: `step_configs/run_train_config/continue_on_num_success`

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: NoneType | float, optional, default: None  
argument path: `step_configs/run_train_config/continue_on_success_ratio`

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: int | NoneType, optional, default: None  
argument path:  
`step_configs/run_train_config/parallelism`

The parallelism for the step

**executor:**

type: dict | NoneType, optional, default: None  
argument path:  
`step_configs/run_train_config/executor`

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)  
argument path: `step_configs/run_train_config/executor/type`  
possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to `lebesgue_v2`:

**extra:**

type: dict, optional  
 argument path:  
 step\_configs/run\_train\_config/  
 executor[lebesgue\_v2]/extra

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional  
 argument path: step\_configs/  
 run\_train\_config/  
 executor[lebesgue\_v2]/extra/  
 scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
 argument path: step\_configs/  
 run\_train\_config/  
 executor[lebesgue\_v2]/extra/  
 program\_id

The ID of the program.

**job\_type:**

type: str, optional, default:  
 container  
 argument path: step\_configs/  
 run\_train\_config/  
 executor[lebesgue\_v2]/extra/  
 job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
 argument path: step\_configs/  
 run\_train\_config/  
 executor[lebesgue\_v2]/extra/  
 template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in Lebesgue.

**prep\_explore\_config:**

type: dict, optional, default: {'template\_config': {'image': 'dpotechnology/dpgen2:latest', 'timeout': None, 'retry\_on\_transient\_error': None, 'timeout\_as\_transient\_error': False, 'envs': None}, 'continue\_on\_failed': False, 'continue\_on\_num\_success': None, 'continue\_on\_success\_ratio': None, 'parallelism': None, 'executor': None}

argument path: `step_configs/prepare_explore_config`

Configuration for prepare exploration

**template\_config:**

type: dict, optional, default: `{'image': 'dptechnology/dpgen2:latest'}`

argument path: `step_configs/prepare_explore_config/template_config`

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default: `dptechnology/dpgen2:latest`

argument path: `step_configs/prepare_explore_config/template_config/image`

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None

argument path: `step_configs/prepare_explore_config/template_config/timeout`

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default: None

argument path: `step_configs/prepare_explore_config/template_config/retry_on_transient_error`

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False

argument path: `step_configs/prepare_explore_config/template_config/timeout_as_transient_error`

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default: None

argument path: `step_configs/prepare_explore_config/template_config/envs`

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False  
 argument path: `step_configs/prep_explore_config/continue_on_failed`

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: int | NoneType, optional, default: None  
 argument path: `step_configs/prep_explore_config/continue_on_num_success`

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: NoneType | float, optional, default: None  
 argument path: `step_configs/prep_explore_config/continue_on_success_ratio`

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: int | NoneType, optional, default: None  
 argument path: `step_configs/prep_explore_config/parallelism`

The parallelism for the step

**executor:**

type: dict | NoneType, optional, default: None  
 argument path: `step_configs/prep_explore_config/executor`

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)  
 argument path: `step_configs/prep_explore_config/executor/type`  
 possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to `lebesgue_v2`:

**extra:**

type: dict, optional  
argument path:  
step\_configs/prepare\_explore\_config/  
executor[lebesgue\_v2]/extra

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional  
argument path: step\_configs/  
prepare\_explore\_config/  
executor[lebesgue\_v2]/extra/  
scass\_type

The machine configuration.

**program\_id:**

type: str, optional  
argument path: step\_configs/  
prepare\_explore\_config/  
executor[lebesgue\_v2]/extra/  
program\_id

The ID of the program.

**job\_type:**

type: str, optional, default:  
container  
argument path: step\_configs/  
prepare\_explore\_config/  
executor[lebesgue\_v2]/extra/  
job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
argument path: step\_configs/  
prepare\_explore\_config/  
executor[lebesgue\_v2]/extra/  
template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in Lebesgue.

**run\_explore\_config:**

type: dict, optional, default: {'template\_config': {'image': 'dptechology/dpgen2:latest', 'timeout': None, 'retry\_on\_transient\_error': None, 'timeout\_as\_transient\_error': False, 'envs': None}, 'continue\_on\_failed': False, 'continue\_on\_num\_success': None, 'continue\_on\_success\_ratio': None, 'parallelism': None, 'executor': None}



argument path: `step_configs/run_explore_config`

Configuration for run exploration

**template\_config:**

type: dict, optional, default: `{'image': 'dptechnology/dpgen2:latest'}`  
argument path: `step_configs/run_explore_config/template_config`

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default: `dptechnology/dpgen2:latest`  
argument path: `step_configs/run_explore_config/template_config/image`

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None  
argument path: `step_configs/run_explore_config/template_config/timeout`

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default: None  
argument path: `step_configs/run_explore_config/template_config/retry_on_transient_error`

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False  
argument path: `step_configs/run_explore_config/template_config/timeout_as_transient_error`

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default: None  
argument path: `step_configs/run_explore_config/template_config/envs`

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False  
argument path: step\_configs/run\_explore\_config/  
continue\_on\_failed

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: int | NoneType, optional, default: None  
argument path: step\_configs/run\_explore\_config/  
continue\_on\_num\_success

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: NoneType | float, optional, default: None  
argument path: step\_configs/run\_explore\_config/  
continue\_on\_success\_ratio

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: int | NoneType, optional, default: None  
argument path:  
step\_configs/run\_explore\_config/parallelism

The parallelism for the step

**executor:**

type: dict | NoneType, optional, default: None  
argument path:  
step\_configs/run\_explore\_config/executor

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)  
argument path: step\_configs/  
run\_explore\_config/executor/type  
possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to [lebesgue\\_v2](#):

**extra:**

type: dict, optional

argument path:  
 step\_configs/run\_explore\_config/  
 executor[lebesgue\_v2]/extra

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional  
 argument path: step\_configs/  
 run\_explore\_config/  
 executor[lebesgue\_v2]/extra/  
 scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
 argument path: step\_configs/  
 run\_explore\_config/  
 executor[lebesgue\_v2]/extra/  
 program\_id

The ID of the program.

**job\_type:**

type: str, optional, default:  
 container  
 argument path: step\_configs/  
 run\_explore\_config/  
 executor[lebesgue\_v2]/extra/  
 job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
 argument path: step\_configs/  
 run\_explore\_config/  
 executor[lebesgue\_v2]/extra/  
 template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in Lebesgue.

**prep\_fp\_config:**

type: dict, optional, default: {'template\_config': {'image': 'dptechnology/dpgen2:latest', 'timeout': None, 'retry\_on\_transient\_error': None, 'timeout\_as\_transient\_error': False, 'envs': None}, 'continue\_on\_failed': False, 'continue\_on\_num\_success': None, 'continue\_on\_success\_ratio': None, 'parallelism': None, 'executor': None}

argument path: step\_configs/prep\_fp\_config

Configuration for prepare fp

**template\_config:**

type: dict, optional, default: {'image':  
'dptechnology/dpgen2:latest'}

argument path:

step\_configs/prep\_fp\_config/template\_config

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default:

dptechnology/dpgen2:latest

argument path:

step\_configs/prep\_fp\_config/  
template\_config/image

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None

argument path:

step\_configs/prep\_fp\_config/  
template\_config/timeout

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default:  
None

argument path: step\_configs/  
prep\_fp\_config/template\_config/  
retry\_on\_transient\_error

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False

argument path: step\_configs/  
prep\_fp\_config/template\_config/  
timeout\_as\_transient\_error

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default:  
None

argument path: step\_configs/  
prep\_fp\_config/template\_config/envs

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False

argument path:

`step_configs/prep_fp_config/continue_on_failed`

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

#### **continue\_on\_num\_success:**

type: `int | NoneType`, optional, default: `None`

argument path: `step_configs/prep_fp_config/continue_on_num_success`

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

#### **continue\_on\_success\_ratio:**

type: `NoneType | float`, optional, default: `None`

argument path: `step_configs/prep_fp_config/continue_on_success_ratio`

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

#### **parallelism:**

type: `int | NoneType`, optional, default: `None`

argument path:

`step_configs/prep_fp_config/parallelism`

The parallelism for the step

#### **executor:**

type: `dict | NoneType`, optional, default: `None`

argument path:

`step_configs/prep_fp_config/executor`

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

##### **type:**

type: `str` (flag key)

argument path: `step_configs/`  
`prep_fp_config/executor/type`

possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to `lebesgue_v2`:

##### **extra:**

type: `dict`, optional

argument path:

`step_configs/prep_fp_config/`  
`executor[lebesgue_v2]/extra`

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional  
argument path:  
step\_configs/prep\_fp\_config/  
executor[lebesgue\_v2]/extra/  
scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
argument path:  
step\_configs/prep\_fp\_config/  
executor[lebesgue\_v2]/extra/  
program\_id

The ID of the program.

**job\_type:**

type: str, optional, default:  
container  
argument path:  
step\_configs/prep\_fp\_config/  
executor[lebesgue\_v2]/extra/  
job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
argument path:  
step\_configs/prep\_fp\_config/  
executor[lebesgue\_v2]/extra/  
template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in Lebesgue.

**run\_fp\_config:**

type: dict, optional, default: {'template\_config': {'image':  
'dptechnology/dpgen2:latest', 'timeout': None,  
'retry\_on\_transient\_error': None,  
'timeout\_as\_transient\_error': False, 'envs': None},  
'continue\_on\_failed': False, 'continue\_on\_num\_success':  
None, 'continue\_on\_success\_ratio': None, 'parallelism':  
None, 'executor': None}

argument path: step\_configs/run\_fp\_config

Configuration for run fp

**template\_config:**

type: dict, optional, default: {'image':  
'dptechnology/dpgen2:latest'}

argument path:  
 step\_configs/run\_fp\_config/template\_config

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default:  
 dptechnology/dpgen2:latest  
 argument path: step\_configs/  
 run\_fp\_config/template\_config/image

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None  
 argument path:  
 step\_configs/run\_fp\_config/  
 template\_config/timeout

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default:  
 None  
 argument path: step\_configs/  
 run\_fp\_config/template\_config/  
 retry\_on\_transient\_error

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False  
 argument path: step\_configs/  
 run\_fp\_config/template\_config/  
 timeout\_as\_transient\_error

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default:  
 None  
 argument path: step\_configs/  
 run\_fp\_config/template\_config/envs

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False  
 argument path:  
 step\_configs/run\_fp\_config/continue\_on\_failed

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: `int | NoneType`, optional, default: `None`

argument path: `step_configs/run_fp_config/continue_on_num_success`

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: `NoneType | float`, optional, default: `None`

argument path: `step_configs/run_fp_config/continue_on_success_ratio`

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: `int | NoneType`, optional, default: `None`

argument path:

`step_configs/run_fp_config/parallelism`

The parallelism for the step

**executor:**

type: `dict | NoneType`, optional, default: `None`

argument path:

`step_configs/run_fp_config/executor`

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: `str` (flag key)

argument path: `step_configs/`

`run_fp_config/executor/type`

possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to `lebesgue_v2`:

**extra:**

type: `dict`, optional

argument path:

`step_configs/run_fp_config/`

`executor[lebesgue_v2]/extra`

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.



**scass\_type:**

type: str, optional  
 argument path:  
 step\_configs/run\_fp\_config/  
 executor[lebesgue\_v2]/extra/  
 scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
 argument path:  
 step\_configs/run\_fp\_config/  
 executor[lebesgue\_v2]/extra/  
 program\_id

The ID of the program.

**job\_type:**

type: str, optional, default:  
 container  
 argument path:  
 step\_configs/run\_fp\_config/  
 executor[lebesgue\_v2]/extra/  
 job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
 argument path:  
 step\_configs/run\_fp\_config/  
 executor[lebesgue\_v2]/extra/  
 template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in  
 Lebesgue.

**select\_confs\_config:**

type: dict, optional, default: {'template\_config': {'image':  
 'dptechnology/dpgen2:latest', 'timeout': None,  
 'retry\_on\_transient\_error': None,  
 'timeout\_as\_transient\_error': False, 'envs': None},  
 'continue\_on\_failed': False, 'continue\_on\_num\_success':  
 None, 'continue\_on\_success\_ratio': None, 'parallelism':  
 None, 'executor': None}

argument path: step\_configs/select\_confs\_config

Configuration for the select confs

**template\_config:**

type: dict, optional, default: {'image':  
 'dptechnology/dpgen2:latest'}  
 argument path: step\_configs/select\_confs\_config/  
 template\_config

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default:  
dptechnology/dpgen2:latest  
argument path:  
step\_configs/select\_confs\_config/  
template\_config/image

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None  
argument path:  
step\_configs/select\_confs\_config/  
template\_config/timeout

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default:  
None  
argument path:  
step\_configs/select\_confs\_config/  
template\_config/  
retry\_on\_transient\_error

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False  
argument path:  
step\_configs/select\_confs\_config/  
template\_config/  
timeout\_as\_transient\_error

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default:  
None  
argument path:  
step\_configs/select\_confs\_config/  
template\_config/envs

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False  
argument path: step\_configs/select\_confs\_config/  
continue\_on\_failed

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: `int | NoneType`, optional, default: `None`

argument path: `step_configs/select_confs_config/continue_on_num_success`

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: `NoneType | float`, optional, default: `None`

argument path: `step_configs/select_confs_config/continue_on_success_ratio`

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: `int | NoneType`, optional, default: `None`

argument path:

`step_configs/select_confs_config/parallelism`

The parallelism for the step

**executor:**

type: `dict | NoneType`, optional, default: `None`

argument path:

`step_configs/select_confs_config/executor`

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: `str` (flag key)

argument path: `step_configs/`

`select_confs_config/executor/type`

possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to `lebesgue_v2`:

**extra:**

type: `dict`, optional

argument path:

`step_configs/select_confs_config/`

`executor[lebesgue_v2]/extra`

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional  
argument path: step\_configs/  
select\_confs\_config/  
executor[lebesgue\_v2]/extra/  
scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
argument path: step\_configs/  
select\_confs\_config/  
executor[lebesgue\_v2]/extra/  
program\_id

The ID of the program.

**job\_type:**

type: str, optional, default:  
container  
argument path: step\_configs/  
select\_confs\_config/  
executor[lebesgue\_v2]/extra/  
job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
argument path: step\_configs/  
select\_confs\_config/  
executor[lebesgue\_v2]/extra/  
template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in  
Lebesgue.

**collect\_data\_config:**

type: dict, optional, default: {'template\_config': {'image':  
'dptechnology/dpgen2:latest', 'timeout': None,  
'retry\_on\_transient\_error': None,  
'timeout\_as\_transient\_error': False, 'envs': None},  
'continue\_on\_failed': False, 'continue\_on\_num\_success':  
None, 'continue\_on\_success\_ratio': None, 'parallelism':  
None, 'executor': None}

argument path: step\_configs/collect\_data\_config

Configuration for the collect data

**template\_config:**

type: dict, optional, default: {'image':  
'dptechnology/dpgen2:latest'}  
argument path: step\_configs/collect\_data\_config/  
template\_config

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default:  
dptechnology/dpgen2:latest  
argument path:  
step\_configs/collect\_data\_config/  
template\_config/image

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None  
argument path:  
step\_configs/collect\_data\_config/  
template\_config/timeout

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default:  
None  
argument path:  
step\_configs/collect\_data\_config/  
template\_config/  
retry\_on\_transient\_error

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False  
argument path:  
step\_configs/collect\_data\_config/  
template\_config/  
timeout\_as\_transient\_error

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default:  
None  
argument path:  
step\_configs/collect\_data\_config/  
template\_config/envs

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False  
argument path: step\_configs/collect\_data\_config/  
continue\_on\_failed

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: int | NoneType, optional, default: None

argument path: step\_configs/collect\_data\_config/  
continue\_on\_num\_success

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: NoneType | float, optional, default: None

argument path: step\_configs/collect\_data\_config/  
continue\_on\_success\_ratio

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: int | NoneType, optional, default: None

argument path:  
step\_configs/collect\_data\_config/parallelism

The parallelism for the step

**executor:**

type: dict | NoneType, optional, default: None

argument path:  
step\_configs/collect\_data\_config/executor

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)

argument path: step\_configs/  
collect\_data\_config/executor/type  
possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to [lebesgue\\_v2](#):

**extra:**

type: dict, optional

argument path:  
step\_configs/collect\_data\_config/  
executor[lebesgue\_v2]/extra

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional  
 argument path: step\_configs/  
 collect\_data\_config/  
 executor[lebesgue\_v2]/extra/  
 scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
 argument path: step\_configs/  
 collect\_data\_config/  
 executor[lebesgue\_v2]/extra/  
 program\_id

The ID of the program.

**job\_type:**

type: str, optional, default:  
 container  
 argument path: step\_configs/  
 collect\_data\_config/  
 executor[lebesgue\_v2]/extra/  
 job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
 argument path: step\_configs/  
 collect\_data\_config/  
 executor[lebesgue\_v2]/extra/  
 template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in  
 Lebesgue.

**cl\_step\_config:**

type: dict, optional, default: {'template\_config': {'image':  
 'dptechnology/dpgen2:latest', 'timeout': None,  
 'retry\_on\_transient\_error': None,  
 'timeout\_as\_transient\_error': False, 'envs': None},  
 'continue\_on\_failed': False, 'continue\_on\_num\_success':  
 None, 'continue\_on\_success\_ratio': None, 'parallelism':  
 None, 'executor': None}

argument path: step\_configs/cl\_step\_config

Configuration for the concurrent learning step

**template\_config:**

type: dict, optional, default: {'image':  
 'dptechnology/dpgen2:latest'}  
 argument path:  
 step\_configs/cl\_step\_config/template\_config

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default:  
dptechnology/dpgen2:latest  
argument path:  
step\_configs/cl\_step\_config/  
template\_config/image

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None  
argument path:  
step\_configs/cl\_step\_config/  
template\_config/timeout

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default:  
None  
argument path: step\_configs/  
cl\_step\_config/template\_config/  
retry\_on\_transient\_error

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False  
argument path: step\_configs/  
cl\_step\_config/template\_config/  
timeout\_as\_transient\_error

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default:  
None  
argument path: step\_configs/  
cl\_step\_config/template\_config/envs

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False  
argument path:  
step\_configs/cl\_step\_config/continue\_on\_failed

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: int | NoneType, optional, default: None



argument path: `step_configs/cl_step_config/continue_on_num_success`

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: `NoneType | float`, optional, default: `None`

argument path: `step_configs/cl_step_config/continue_on_success_ratio`

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: `int | NoneType`, optional, default: `None`

argument path: `step_configs/cl_step_config/parallelism`

The parallelism for the step

**executor:**

type: `dict | NoneType`, optional, default: `None`

argument path: `step_configs/cl_step_config/executor`

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: `str` (flag key)

argument path: `step_configs/cl_step_config/executor/type`

possible choices: [lebesgue\\_v2](#)

The type of the executor.

When *type* is set to `lebesgue_v2`:

**extra:**

type: `dict`, optional

argument path: `step_configs/cl_step_config/executor[lebesgue_v2]/extra`

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: `str`, optional

argument path:  
step\_configs/cl\_step\_config/  
executor[lebesgue\_v2]/extra/  
scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
argument path:  
step\_configs/cl\_step\_config/  
executor[lebesgue\_v2]/extra/  
program\_id

The ID of the program.

**job\_type:**

type: str, optional, default:  
container  
argument path:  
step\_configs/cl\_step\_config/  
executor[lebesgue\_v2]/extra/  
job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
argument path:  
step\_configs/cl\_step\_config/  
executor[lebesgue\_v2]/extra/  
template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in  
Lebesgue.

**default\_step\_config:**

type: dict, optional, default: {}  
argument path: default\_step\_config

The default step configuration.

**template\_config:**

type: dict, optional, default: {'image':  
'dptechnology/dpgen2:latest'}  
argument path: default\_step\_config/template\_config

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default:  
dptechnology/dpgen2:latest  
argument path:  
default\_step\_config/template\_config/image

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None

argument path:

default\_step\_config/template\_config/timeout

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default: None

argument path: default\_step\_config/

template\_config/retry\_on\_transient\_error

Retry the step if a TransientError is raised.

**timeout\_as\_transient\_error:**

type: bool, optional, default: False

argument path: default\_step\_config/

template\_config/timeout\_as\_transient\_error

Treat the timeout as TransientError.

**envs:**

type: dict | NoneType, optional, default: None

argument path:

default\_step\_config/template\_config/envs

The environmental variables.

**continue\_on\_failed:**

type: bool, optional, default: False

argument path: default\_step\_config/continue\_on\_failed

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**continue\_on\_num\_success:**

type: int | NoneType, optional, default: None

argument path: default\_step\_config/continue\_on\_num\_success

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_success\_ratio:**

type: NoneType | float, optional, default: None

argument path: default\_step\_config/continue\_on\_success\_ratio

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**parallelism:**

type: int | NoneType, optional, default: None

argument path: default\_step\_config/parallelism

The parallelism for the step

**executor:**

type: dict | NoneType, optional, default: None  
argument path: default\_step\_config/executor

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)  
argument path: default\_step\_config/executor/type  
possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to [lebesgue\\_v2](#):

**extra:**

type: dict, optional  
argument path: default\_step\_config/  
executor[lebesgue\_v2]/extra

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional  
argument path: default\_step\_config/  
executor[lebesgue\_v2]/extra/  
scass\_type

The machine configuraiton.

**program\_id:**

type: str, optional  
argument path: default\_step\_config/  
executor[lebesgue\_v2]/extra/  
program\_id

The ID of the program.

**job\_type:**

type: str, optional, default: container  
argument path: default\_step\_config/  
executor[lebesgue\_v2]/extra/  
job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
argument path: default\_step\_config/  
executor[lebesgue\_v2]/extra/  
template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in Lebesgue.

**lebesgue\_context\_config:**

type: dict | NoneType, optional, default: None

argument path: lebesgue\_context\_config

Configuration passed to dflow Lebesgue context

**s3\_config:**

type: dict | NoneType, optional, default: None

argument path: s3\_config

The S3 configuration passed to dflow

**dflow\_config:**

type: dict | NoneType, optional, default: None

argument path: dflow\_config

The configuration passed to dflow



## OP CONFIGS

### 5.1 RunDPTrain

**init\_model\_start\_pref\_v:**

type: float, optional, default: 0.0  
argument path: init\_model\_start\_pref\_v  
The start virial prefactor in loss when init-model

**init\_model\_start\_pref\_f:**

type: int | float, optional, default: 100  
argument path: init\_model\_start\_pref\_f  
The start force prefactor in loss when init-model

**init\_model\_start\_pref\_e:**

type: float, optional, default: 0.1  
argument path: init\_model\_start\_pref\_e  
The start energy prefactor in loss when init-model

**init\_model\_start\_lr:**

type: float, optional, default: 0.0001  
argument path: init\_model\_start\_lr  
The start learning rate when init-model

**init\_model\_numb\_steps:**

type: int, optional, default: 400000, alias: *init\_model\_stop\_batch*  
argument path: init\_model\_numb\_steps  
The number of training steps when init-model

**init\_model\_old\_ratio:**

type: float, optional, default: 0.9  
argument path: init\_model\_old\_ratio  
The frequency ratio of old data over new data

**init\_model\_policy:**

type: str, optional, default: no  
argument path: init\_model\_policy  
The policy of init-model training. It can be

- ‘no’: No init-model training. Traing from scratch.

- ‘yes’: Do init-model training.
- ‘old\_data\_larger\_than:XXX’: Do init-model if the training data size of the previous model is larger than XXX. XXX is an int number.

## 5.2 RunLmp

### **command:**

type: str, optional, default: lmp

argument path: command

The command of LAMMPS

## 5.3 RunVasp

### **out:**

type: str, optional, default: data

argument path: out

The output dir name of labeled data. In *deepmd/npz* format provided by *dpdata*.

### **log:**

type: str, optional, default: vasp.log

argument path: log

The log file name of VASP

### **command:**

type: str, optional, default: vasp

argument path: command

The command of VASP



## ALLOY CONFIGS

**fmt:**

type: str, optional, default: lammmps/lmp

argument path: `fmt`

The format of file content

**atom\_pert\_dist:**

type: float, optional, default: 0.0

argument path: `atom_pert_dist`

The distance of atomic position perturbation

**cell\_pert\_frac:**

type: float, optional, default: 0.0

argument path: `cell_pert_frac`

The fraction of cell perturbation

**concentration:**

type: list | NoneType, optional, default: None

argument path: `concentration`

The concentration of each element. If None all elements have the same concentration

**numb\_confs:**

type: int, optional, default: 1

argument path: `numb_confs`

The number of configurations to generate

**replicate:**

type: list | NoneType, optional, default: None

argument path: `replicate`

The number of replicates in each direction

**type\_map:**

type: list

argument path: `type_map`

The type map of the system

**lattice:**

type: list | tuple

argument path: `lattice`

The lattice. Should be a list providing [ “lattice\_type”, lattice\_const ], or a list providing [ “/path/to/dpdata/system”, “fmt” ]. The two styles are distinguished by the type of the second element.

## STEP CONFIGS

### 7.1 Configurations for dflow steps

**executor:**

type: dict | NoneType, optional, default: None

argument path: `executor`

The executor of the step.

Depending on the value of *type*, different sub args are accepted.

**type:**

type: str (flag key)

argument path: `executor/type`

possible choices: [lebesgue\\_v2](#)

The type of the executor.

When [type](#) is set to `lebesgue_v2`:

**extra:**

type: dict, optional

argument path: `executor[lebesgue_v2]/extra`

The 'extra' key in the lebesgue executor. Note that we do not check if 'the *dict* provided to the 'extra' key is valid or not.

**scass\_type:**

type: str, optional

argument path:

`executor[lebesgue_v2]/extra/scass_type`

The machine configuraiton.

**program\_id:**

type: str, optional

argument path:

`executor[lebesgue_v2]/extra/program_id`

The ID of the program.

**job\_type:**

type: str, optional, default: container

argument path:  
executor[lebesgue\_v2]/extra/job\_type

The type of job.

**template\_cover\_cmd\_escape\_bug:**

type: bool, optional, default: True  
argument path: executor[lebesgue\_v2]/extra/  
template\_cover\_cmd\_escape\_bug

The key for hacking around a bug in Lebesgue.

**parallelism:**

type: int | NoneType, optional, default: None  
argument path: parallelism

The parallelism for the step

**continue\_on\_success\_ratio:**

type: NoneType | float, optional, default: None  
argument path: continue\_on\_success\_ratio

Only in the sliced OP case. Continue the workflow if a certain ratio of the sliced jobs are successful.

**continue\_on\_num\_success:**

type: int | NoneType, optional, default: None  
argument path: continue\_on\_num\_success

Only in the sliced OP case. Continue the workflow if a certain number of the sliced jobs are successful.

**continue\_on\_failed:**

type: bool, optional, default: False  
argument path: continue\_on\_failed

If continue the the step is failed (FatalError, TransientError, A certain number of retrial is reached...).

**template\_config:**

type: dict, optional, default: {'image': 'dptechnology/dpgen2:latest'}  
argument path: template\_config

The configs passed to the PythonOPTemplate.

**image:**

type: str, optional, default: dptechnology/dpgen2:latest  
argument path: template\_config/image

The image to run the step.

**timeout:**

type: int | NoneType, optional, default: None  
argument path: template\_config/timeout

The time limit of the OP. Unit is second.

**retry\_on\_transient\_error:**

type: NoneType | bool, optional, default: None

argument path: `template_config/retry_on_transient_error`

Retry the step if a `TransientError` is raised.

**timeout\_as\_transient\_error:**

type: `bool`, optional, default: `False`

argument path: `template_config/timeout_as_transient_error`

Treat the timeout as `TransientError`.

**envs:**

type: `dict | NoneType`, optional, default: `None`

argument path: `template_config/envs`

The environmental variables.



## DEVELOPERS' GUIDE

- The concurrent learning algorithm
- Overview of the DPGEN2 implementation
- The DPGEN2 workflow
- How to contribute

### 8.1 The concurrent learning algorithm

DPGEN2 implements the concurrent learning algorithm named DP-GEN, described in [this paper](#). It is noted that other types of workflows, like active learning, should be easily implemented within the infrastructure of DPGEN2.

The DP-GEN algorithm is iterative. In each iteration, four steps are consecutively executed: training, exploration, selection, and labeling.

1. **Training.** A set of DP models are trained with the same dataset and the same hyperparameters. The only difference is the random seed initializing the model parameters.
2. **Exploration.** One of the DP models is used to explore the configuration space. The strategy of exploration highly depends on the purpose of the application case of the model. The simulation technique for exploration can be molecular dynamics, Monte Carlo, structure search/optimization, enhanced sampling, or any combination of them. Current DPGEN2 only supports exploration based on molecular simulation platform [LAMMPS](#).
3. **Selection.** Not all the explored configurations are labeled, rather, the model prediction errors on the configurations are estimated by the *model deviation*, which is defined as the standard deviation in predictions of the set of the models. The critical configurations with large and not-that-large errors are selected for labeling. The configurations with very large errors are not selected because the large error is usually caused by non-physical configurations, e.g. overlapping atoms.
4. **Labeling.** The selected configurations are labeled with energy, forces, and virial calculated by a method of first-principles accuracy. The usually used method is the [density functional theory](#) implemented in [VASP](#), [Quantum Espresso](#), [CP2K](#), and etc.. The labeled data are finally added to the training dataset to start the next iteration.

In each iteration, the quality of the model is improved by selecting and labeling more critical data and adding them to the training dataset. The DP-GEN iteration is converged when no more critical data can be selected.

## 8.2 Overview of the DPGEN2 Implementation

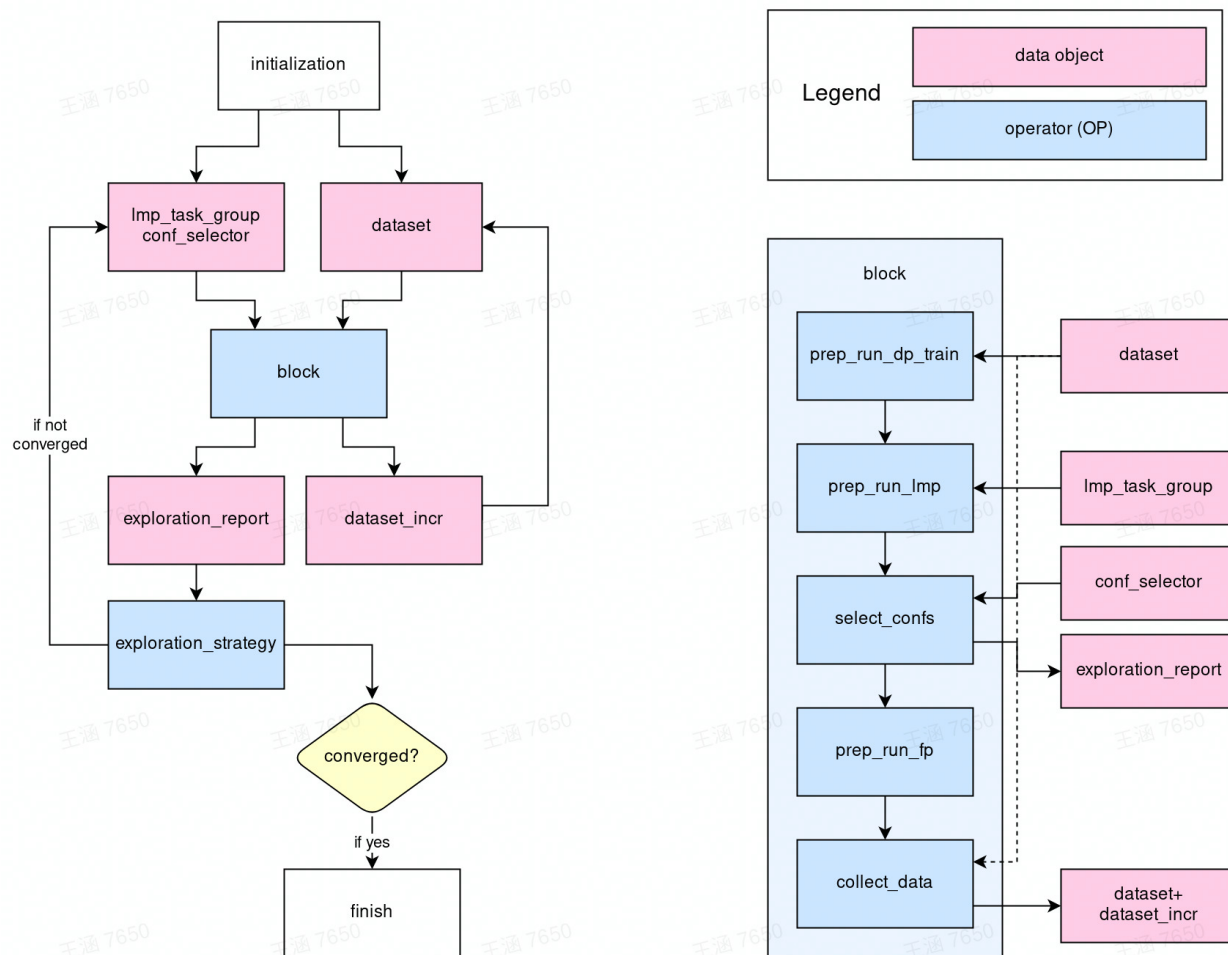
The implementation DPGEN2 is based on the workflow platform [dflow](#), which is a python wrapper of the [Argo Workflows](#), an open-source container-native workflow engine on [Kubernetes](#).

The DP-GEN algorithm is conceptually modeled as a computational graph. The implementation is then considered as two lines: the operators and the workflow.

1. **Operators.** Operators are implemented in Python v3. The operators should be implemented and tested *without* the workflow.
2. **Workflow.** Workflow is implemented on [dflow](#). Ideally, the workflow is implemented and tested with all operators mocked.

## 8.3 The DPGEN2 workflow

The workflow of DPGEN2 is illustrated in the following figure



In the center is the block operator, which is a super-OP (an OP composed by several OPs) for one DP-GEN iteration, i.e. the super-OP of the training, exploration, selection, and labeling steps. The inputs of the block OP are `Imp_task_group`, `conf_selector` and `dataset`.

- `Imp_task_group`: definition of a group of LAMMPS tasks that explore the configuration space.



- `conf_selector`: defines the rule by which the configurations are selected for labeling.
- `dataset`: the training dataset.

The outputs of the block OP are

- `exploration_report`: a report recording the result of the exploration. For example, how many configurations are accurate enough and how many are selected as candidates for labeling.
- `dataset_incr`: the increment of the training dataset.

The `dataset_incr` is added to the training dataset.

The `exploration_report` is passed to the `exploration_strategy` OP. The `exploration_strategy` implements the strategy of exploration. It reads the `exploration_report` generated by each iteration (block), then tells if the iteration is converged. If not, it generates a group of LAMMPS tasks (`lmp_task_group`) and the criteria of selecting configurations (`conf_selector`). The `lmp_task_group` and `conf_selector` are then used by block of the next iteration. The iteration closes.

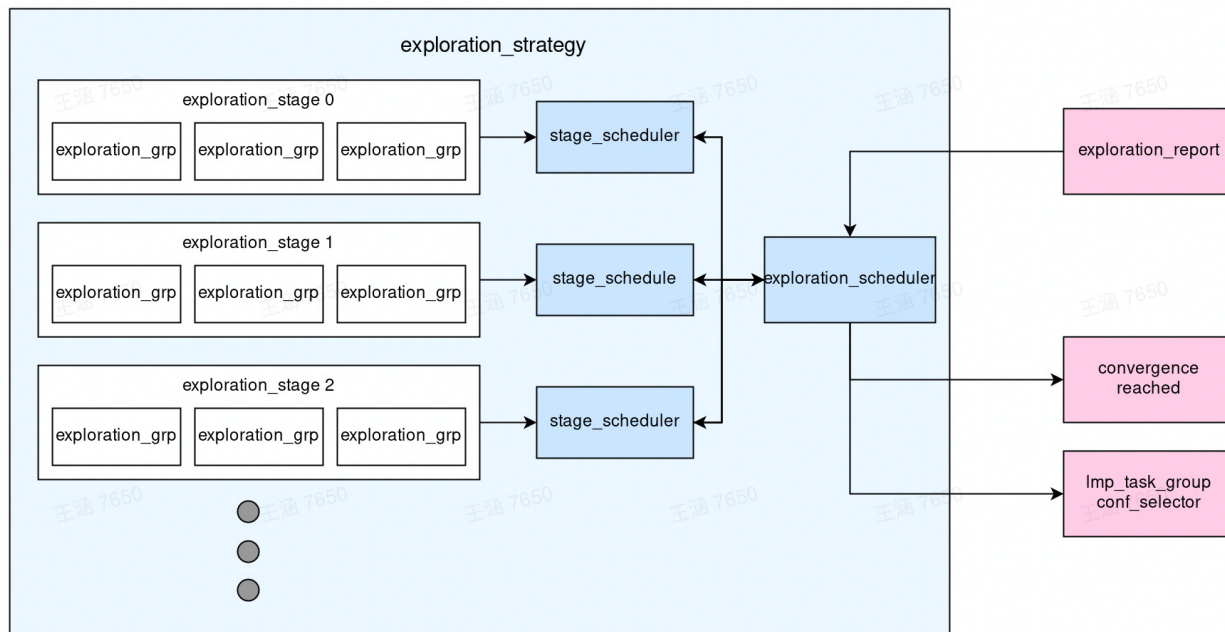
### 8.3.1 Inside the block operator

The inside of the super-OP block is displayed on the right-hand side of the figure. It contains the following steps to finish one DPGEN iteration

- `prep_run_dp_train`: prepares training tasks of DP models and runs them.
- `prep_run_lmp`: prepares the LAMMPS exploration tasks and runs them.
- `select_confs`: selects configurations for labeling from the explored configurations.
- `prep_run_fp`: prepares and runs first-principles tasks.
- `collect_data`: collects the `dataset_incr` and adds it to the dataset.

### 8.3.2 The exploration strategy

The exploration strategy defines how the configuration space is explored by the concurrent learning algorithm. The design of the exploration strategy is graphically illustrated in the following figure. The exploration is composed of stages. Only the DP-GEN exploration is converged at one stage (no configuration with a large error is explored), the exploration goes to the next iteration. The whole procedure is controlled by `exploration_scheduler`. Each stage has its schedule, which talks to the `exploration_scheduler` to generate the schedule for the DP-GEN algorithm.



Some concepts are explained below:

- **Exploration group.** A group of LAMMPS tasks shares similar settings. For example, a group of NPT MD simulations in a certain thermodynamic space.
- **Exploration stage.** The `exploration_stage` contains a list of exploration groups. It contains all information needed to define the `lmp_task_group` used by the block in the DP-GEN iteration.
- **Stage scheduler.** It guarantees the convergence of the DP-GEN algorithm in each `exploration_stage`. If the exploration is not converged, the `stage_scheduler` generates `lmp_task_group` and `conf_selector` from the `exploration_stage` for the next iteration (probably with a different initial condition, i.e. different initial configurations and randomly generated initial velocity).
- **Exploration scheduler.** The scheduler for the DP-GEN algorithm. When DP-GEN is converged in one of the stages, it goes to the next stage until all planned stages are used.

## 8.4 How to contribute

Anyone interested in the DPGEN2 project may contribute OPs, workflows, and exploration strategies.

- To contribute OPs, one may check the [guide on writing operators](#)
- To contribute workflows, one may take the DP-GEN workflow as an example. It is implemented in `dp-gen2/flow/dpgen_loop.py` and tested with all operators mocked in `test/test_dpgen_loop.py`
- To contribute the exploration strategy, one may check the [guide on writing exploration strategies](#)

## OPERATORS

There are two types of OPs in DPGEN2

- OP. An execution unit the the workflow. It can be roughly viewed as a piece of Python script taking some input and gives some outputs. An OP cannot be used in the `dflow` until it is embedded in a super-OP.
- Super-OP. An execution unite that is composed by one or more OP and/or super-OPs.

Technically, OP is a Python class derived from `dflow.python.OP`. It serves as the `PythonOPTemplate` of `dflow.Step`.

The super-OP is a Python class derived from `dflow.Steps`. It contains `dflow.Steps` as building blocks, and can be used as OP template to generate a `dflow.Step`. The explanation of the concepts `dflow.Step` and `dflow.Steps`, one may refer to the [manual of dflow](#).

### 9.1 The super-OP `PrepRunDPTrain`

In the following we will take the `PrepRunDPTrain` super-OP as an example to illustrate how to write OPs in DPGEN2.

`PrepRunDPTrain` is a super-OP that prepares several DeePMD-kit training tasks, and submit all of them. This super-OP is composed by two `dflow.Steps` building from `dflow.python.OPs` `PrepDPTrain` and `RunDPTrain`.

```
from dflow import (
    Step,
    Steps,
)
from dflow.python import(
    PythonOPTemplate,
    OP,
    Slices,
)

class PrepRunDPTrain(Steps):
    def __init__(
        self,
        name : str,
        prep_train_op : OP,
        run_train_op : OP,
        prep_train_image : str = "dflow:v1.0",
        run_train_image : str = "dflow:v1.0",
    ):
        ...
```

(continues on next page)

(continued from previous page)

```

self = _prep_run_dp_train(
    self,
    self.step_keys,
    prep_train_op,
    run_train_op,
    prep_train_image = prep_train_image,
    run_train_image = run_train_image,
)

```

The construction of the `PrepRunDPTrain` takes prepare-training OP and run-training OP and their docker images as input, and implemented in internal method `_prep_run_dp_train`.

```

def _prep_run_dp_train(
    train_steps,
    step_keys,
    prep_train_op : OP = PrepDPTrain,
    run_train_op : OP = RunDPTrain,
    prep_train_image : str = "dflow:v1.0",
    run_train_image : str = "dflow:v1.0",
):
    prep_train = Step(
        ...
        template=PythonOPTemplate(
            prep_train_op,
            image=prep_train_image,
            ...
        ),
        ...
    )
    train_steps.add(prepare_train)

    run_train = Step(
        ...
        template=PythonOPTemplate(
            run_train_op,
            image=run_train_image,
            ...
        ),
        ...
    )
    train_steps.add(run_train)

    train_steps.outputs.artifacts["scripts"]._from = run_train.outputs.artifacts["script
↪"]
    train_steps.outputs.artifacts["models"]._from = run_train.outputs.artifacts["model"]
    train_steps.outputs.artifacts["logs"]._from = run_train.outputs.artifacts["log"]
    train_steps.outputs.artifacts["lcurves"]._from = run_train.outputs.artifacts["lcurve
↪"]

    return train_steps

```

In `_prep_run_dp_train`, two instances of `dflow.Step`, i.e. `prep_train` and `run_train`, generated from `prep_train_op` and `run_train_op`, respectively, are added to `train_steps`. Both of `prep_train_op` and

`run_train_op` are OPs (python classes derived from `dflow.python.OPs`) that will be illustrated later. `train_steps` is an instance of `dflow.Steps`. The outputs of the second OP `run_train` are assigned to the outputs of the `train_steps`.

The `prep_train` prepares a list of paths, each of which contains all necessary files to start a DeePMD-kit training tasks.

The `run_train` slices the list of paths, and assign each item in the list to a DeePMD-kit task. The task is executed by `run_train_op`. This is a very nice feature of `dflow`, because the developer only needs to implement how one DeePMD-kit task is executed, and then all the items in the task list will be executed **in parallel**. See the following code to see how it works

```
run_train = Step(
    'run-train',
    template=PythonOPTemplate(
        run_train_op,
        image=run_train_image,
        slices = Slices(
            "int('{{item}}')",
            input_parameter = ["task_name"],
            input_artifact = ["task_path", "init_model"],
            output_artifact = ["model", "lcurve", "log", "script"],
        ),
    ),
    parameters={
        "config" : train_steps.inputs.parameters["train_config"],
        "task_name" : prep_train.outputs.parameters["task_names"],
    },
    artifacts={
        'task_path' : prep_train.outputs.artifacts['task_paths'],
        "init_model" : train_steps.inputs.artifacts['init_models'],
        "init_data": train_steps.inputs.artifacts['init_data'],
        "iter_data": train_steps.inputs.artifacts['iter_data'],
    },
    with_sequence=argo_sequence(argo_len(prep_train.outputs.parameters["task_names
↪"]), format=train_index_pattern),
    key = step_keys['run-train'],
)
```

The input parameter "task\_names" and artifacts "task\_paths" and "init\_model" are sliced and supplied to each DeePMD-kit task. The output artifacts of the tasks ("model", "lcurve", "log" and "script") are stacked in the same order as the input lists. These lists are assigned as the outputs of `train_steps` by

```
train_steps.outputs.artifacts["scripts"]._from = run_train.outputs.artifacts["script
↪"]
train_steps.outputs.artifacts["models"]._from = run_train.outputs.artifacts["model"]
train_steps.outputs.artifacts["logs"]._from = run_train.outputs.artifacts["log"]
train_steps.outputs.artifacts["lcurves"]._from = run_train.outputs.artifacts["lcurve
↪"]
```

## 9.2 The OP RunDPTrain

We will take RunDPTrain as an example to illustrate how to implement an OP in DPGEN2. The source code of this OP is found [here](#)

Firstly of all, an OP should be implemented as a derived class of `dflow.python.OP`.

The `dflow.python.OP` requires static type define for the input and output variables, i.e. the signatures of an OP. The input and output signatures of the `dflow.python.OP` are given by classmethods `get_input_sign` and `get_output_sign`.

```
from dflow.python import (
    OP,
    OPIO,
    OPIOSign,
    Artifact,
)
class RunDPTrain(OP):
    @classmethod
    def get_input_sign(cls):
        return OPIOSign({
            "config" : dict,
            "task_name" : str,
            "task_path" : Artifact(Path),
            "init_model" : Artifact(Path),
            "init_data" : Artifact(List[Path]),
            "iter_data" : Artifact(List[Path]),
        })

    @classmethod
    def get_output_sign(cls):
        return OPIOSign({
            "script" : Artifact(Path),
            "model" : Artifact(Path),
            "lcurve" : Artifact(Path),
            "log" : Artifact(Path),
        })
```

All items not defined as `Artifact` are treated as parameters of the OP. The concept of parameter and artifact are explained in the [dflow document](#). To be short, the artifacts can be `pathlib.Path` or a list of `pathlib.Path`. The artifacts are passed by the file system. Other data structures are treated as parameters, they are passed as variables encoded in `str`. Therefore, a large amount of information should be stored in artifacts, otherwise they can be considered as parameters.

The operation of the OP is implemented in method `execute`, and are run in docker containers. Again taking the `execute` method of `RunDPTrain` as an example

```
@OP.exec_sign_check
def execute(
    self,
    ip : OPIO,
) -> OPIO:
    ...
    task_name = ip['task_name']
```

(continues on next page)

(continued from previous page)

```

task_path = ip['task_path']
init_model = ip['init_model']
init_data = ip['init_data']
iter_data = ip['iter_data']
...
work_dir = Path(task_name)
...
# here copy all files in task_path to work_dir
...
with set_directory(work_dir):
    fplog = open('train.log', 'w')
    def clean_before_quit():
        fplog.close()
    # train model
    command = ['dp', 'train', train_script_name]
    ret, out, err = run_command(command)
    if ret != 0:
        clean_before_quit()
        raise FatalError('dp train failed')
    fplog.write(out)
    # freeze model
    ret, out, err = run_command(['dp', 'freeze', '-o', 'frozen_model.pb'])
    if ret != 0:
        clean_before_quit()
        raise FatalError('dp freeze failed')
    fplog.write(out)
    clean_before_quit()

return OPIO({
    "script" : work_dir / train_script_name,
    "model" : work_dir / "frozen_model.pb",
    "lcurve" : work_dir / "lcurve.out",
    "log" : work_dir / "train.log",
})

```

The inputs and outputs variables are recorded in data structure `dflow.python.OPIO`, which is initialized by a Python dict. The keys in the input/output dict, and the types of the input/output variables will be checked against their signatures by decorator `OP.exec_sign_check`. If any key or type does not match, an exception will be raised.

It is noted that all input artifacts of the OP are read-only, therefore, the first step of the `RunDPTrain.execute` is to copy all necessary input files from the directory `task_path` prepared by `PrepDPTrain` to the working directory `work_dir`.

`with_directory` method creates the `work_dir` and swithes to the directory before the execution, and then exits the directoy when the task finishes or an error is raised.

In what follows, the training and model frozen bash commands are executed consecutively. The return code is check and a `FatalError` is raised if a non-zero code is detected.

Finally the trained model file, input script, learning curve file and the log file are recored in a `dflow.python.OPIO` and returned.





## EXPLORATION

DPGEN2 allows developers to contribute exploration strategies. The exploration strategy defines how the configuration space is explored by molecular simulations in each DPGEN iteration. Notice that we are not restricted to molecular dynamics, any molecular simulation is, in principle, allowed. For example, Monte Carlo, enhanced sampling, structure optimization, and so on.

An exploration strategy takes the history of exploration as input, and gives back DPGEN the exploration tasks (we call it **task group**) and the rule to select configurations from the trajectories generated by the tasks (we call it **configuration selector**).

One can contribute from three aspects:

- The stage scheduler
- The exploration task groups
- Configuration selector

### 10.1 Stage scheduler

The stage scheduler takes an exploration report passed from the exploration scheduler as input, and tells the exploration scheduler if the exploration in the stage is converged, if not, returns a group of exploration tasks and a configuration selector that are used in the next DPGEN iteration.

Detailed explanation of the concepts are found [here](#).

All the stage schedulers are derived from the abstract base class `StageScheduler`. The only interface to be implemented is `StageScheduler.plan_next_iteration`. One may check the doc string for the explanation of the interface.

```
class StageScheduler(ABC):
    """
    The scheduler for an exploration stage.
    """

    @abstractmethod
    def plan_next_iteration(
        self,
        hist_reports : List[ExplorationReport],
        report : ExplorationReport,
        confs : List[Path],
    ) -> Tuple[bool, ExplorationTaskGroup, ConfSelector] :
        """
```

(continues on next page)

(continued from previous page)

```

    Make the plan for the next iteration of the stage.

    It checks the report of the current and all historical iterations of the stage,
    and tells if the iterations are converged.
    If not converged, it will plan the next iteration for the stage.

    Parameters
    -----
    hist_reports: List[ExplorationReport]
        The historical exploration report of the stage. If this is the first
↪ iteration of the stage, this list is empty.
    report : ExplorationReport
        The exploration report of this iteration.
    confs: List[Path]
        A list of configurations generated during the exploration. May be used to
↪ generate new configurations for the next iteration.

    Returns
    -----
    converged: bool
        If the stage converged.
    task: ExplorationTaskGroup
        A `ExplorationTaskGroup` defining the exploration of the next iteration.
↪ Should be `None` if the stage is converged.
    conf_selector: ConfSelector
        The configuration selector for the next iteration. Should be `None` if the
↪ stage is converged.

    """

```

One may check more details on the exploratin task group and the configuration selector.

## 10.2 Exploration task groups

DPGEN2 defines a python class `ExplorationTask` to manage all necessary files needed to run a exploration task. It can be used as the example provided in the doc string.

```

class ExplorationTask():
    """Define the files needed by an exploration task.

    Examples
    -----
    >>> # this example dumps all files needed by the task.
    >>> files = exploration_task.files()
    ... for file_name, file_content in files.items():
    ...     with open(file_name, 'w') as fp:
    ...         fp.write(file_content)

    """

```

A collection of the exploration tasks is called exploration task group. All tasks groups are derived from the base class

ExplorationTaskGroup. The exploration task group can be viewed as a list of ExplorationTasks, one may get the list by using property ExplorationTaskGroup.task\_list. One may add tasks, or ExplorationTaskGroup to the group by methods ExplorationTaskGroup.add\_task and ExplorationTaskGroup.add\_group, respectively.

```
class ExplorationTaskGroup(Sequence):
    @property
    def task_list(self) -> List[ExplorationTask]:
        """Get the `list` of `ExplorationTask`"""
        ...

    def add_task(self, task: ExplorationTask):
        """Add one task to the group."""
        ...

    def add_group(
        self,
        group : 'ExplorationTaskGroup',
    ):
        """Add another group to the group."""
        ...
```

An example of generating a group of NPT MD simulations may illustrate how to implement the ExplorationTaskGroups.

## 10.3 Configuration selector

The configuration selectors are derived from the abstract base class ConfSelector

```
class ConfSelector(ABC):
    """Select configurations from trajectory and model deviation files.
    """
    @abstractmethod
    def select (
        self,
        trajs : List[Path],
        model_devis : List[Path],
        traj_fmt : str = 'deepmd/npz',
        type_map : List[str] = None,
    ) -> Tuple[List[ Path ], ExplorationReport]:
```

The abstractmethod to implement is ConfSelector.select. trajs and model\_devis are lists of files that recording the simulations trajectories and model deviations respectively. traj\_fmt and type\_map are parameters that may be needed for loading the trajectories by dpdata.

The ConfSelector.select returns a Path, each of which can be treated as a dpdata.MultiSystems, and a ExplorationReport.

An example of selecting configurations from LAMMPS trajectories may illustrate how to implement the ConfSelectors.



## DPGEN2 API

### 11.1 dpgen2 package

#### 11.1.1 Subpackages

`dpgen2.entrypoint` package

Submodules

`dpgen2.entrypoint.download` module

`dpgen2.entrypoint.download.download`(*workflow\_id*, *wf\_config*: *Optional*[*Dict*] = {}, *wf\_keys*: *Optional*[*List*] = None, *prefix*: *Optional*[*str*] = None)

`dpgen2.entrypoint.main` module

`dpgen2.entrypoint.main.main`()

`dpgen2.entrypoint.main.main_parser`() → *ArgumentParser*

DPGEN2 commandline options argument parser.

**Returns**

**`argparse.ArgumentParser`**

the argument parser

#### Notes

This function is used by documentation.

`dpgen2.entrypoint.main.parse_args`(*args*: *Optional*[*List*[*str*]] = None)

DPGEN2 commandline options argument parsing.

**Parameters**

**`args`: *List*[*str*]**

list of command line arguments, main purpose is testing default option None takes arguments from `sys.argv`

### **dpgen2.entrypoint.showkey module**

`dpgen2.entrypoint.showkey.showkey(wf_id, wf_config)`

### **dpgen2.entrypoint.status module**

`dpgen2.entrypoint.status.status(workflow_id, wf_config: Optional[Dict] = {})`

### **dpgen2.entrypoint.submit module**

`dpgen2.entrypoint.submit.expand_idx(in_list)`

`dpgen2.entrypoint.submit.expand_sys_str(root_dir: Union[str, Path]) → List[str]`

`dpgen2.entrypoint.submit.get_kspacing_kgamma_from_incar(fname)`

```

dpngen2.entrypoint.submit.make_concurrent_learning_op(train_style: str = 'dp', explore_style: str =
'imp', fp_style: str = 'vasp', prep_train_config:
str = {'continue_on_failed': False,
'continue_on_num_success': None,
'continue_on_success_ratio': None, 'executor':
None, 'parallelism': None, 'template_config':
{'envs': None, 'image':
'dptechnology/dpgen2:latest',
'retry_on_transient_error': None, 'timeout':
None, 'timeout_as_transient_error': False}},
run_train_config: str = {'continue_on_failed':
False, 'continue_on_num_success': None,
'continue_on_success_ratio': None, 'executor':
None, 'parallelism': None, 'template_config':
{'envs': None, 'image':
'dptechnology/dpgen2:latest',
'retry_on_transient_error': None, 'timeout':
None, 'timeout_as_transient_error': False}},
prep_explore_config: str =
{'continue_on_failed': False,
'continue_on_num_success': None,
'continue_on_success_ratio': None, 'executor':
None, 'parallelism': None, 'template_config':
{'envs': None, 'image':
'dptechnology/dpgen2:latest',
'retry_on_transient_error': None, 'timeout':
None, 'timeout_as_transient_error': False}},
run_explore_config: str =
{'continue_on_failed': False,
'continue_on_num_success': None,
'continue_on_success_ratio': None, 'executor':
None, 'parallelism': None, 'template_config':
{'envs': None, 'image':
'dptechnology/dpgen2:latest',
'retry_on_transient_error': None, 'timeout':
None, 'timeout_as_transient_error': False}},
prep_fp_config: str = {'continue_on_failed':
False, 'continue_on_num_success': None,
'continue_on_success_ratio': None, 'executor':
None, 'parallelism': None, 'template_config':
{'envs': None, 'image':
'dptechnology/dpgen2:latest',
'retry_on_transient_error': None, 'timeout':
None, 'timeout_as_transient_error': False}},
run_fp_config: str = {'continue_on_failed':
False, 'continue_on_num_success': None,
'continue_on_success_ratio': None, 'executor':
None, 'parallelism': None, 'template_config':
{'envs': None, 'image':
'dptechnology/dpgen2:latest',
'retry_on_transient_error': None, 'timeout':
None, 'timeout_as_transient_error': False}},
select_confs_config: str =
{'continue_on_failed': False,
'continue_on_num_success': None,
'continue_on_success_ratio': None, 'executor':
None, 'parallelism': None, 'template_config':
{'envs': None, 'image':
'dptechnology/dpgen2:latest',
'retry_on_transient_error': None, 'timeout':

```

```
dpgen2.entrypoint.submit.make_conf_list(conf_list, type_map, fmt='vasp/poscar')
dpgen2.entrypoint.submit.make_naive_exploration_scheduler(config, old_style=False)
dpgen2.entrypoint.submit.print_list_steps(steps)
dpgen2.entrypoint.submit.resubmit_concurrent_learning(wf_config, wfid, list_steps=False,
                                                       reuse=None, old_style=False)
dpgen2.entrypoint.submit.submit_concurrent_learning(wf_config, reuse_step=None, old_style=False)
dpgen2.entrypoint.submit.successful_step_keys(wf)
dpgen2.entrypoint.submit.wf_global_workflow(wf_config)
dpgen2.entrypoint.submit.workflow_concurrent_learning(config: Dict, old_style: Optional[bool] =
                                                       False)
```

### dpgen2.entrypoint.submit\_args module

```
dpgen2.entrypoint.submit_args.default_step_config_args()
dpgen2.entrypoint.submit_args.dflow_conf_args()
dpgen2.entrypoint.submit_args.dp_train_args()
dpgen2.entrypoint.submit_args.dpgen_step_config_args(default_config)
dpgen2.entrypoint.submit_args.gen_doc(*, make_anchor=True, make_link=True, **kwargs)
dpgen2.entrypoint.submit_args.input_args()
dpgen2.entrypoint.submit_args.lebesgue_conf_args()
dpgen2.entrypoint.submit_args.lmp_args()
dpgen2.entrypoint.submit_args.normalize(data)
dpgen2.entrypoint.submit_args.submit_args(default_step_config={'continue_on_failed': False,
                                                                'continue_on_num_success': None,
                                                                'continue_on_success_ratio': None, 'executor': None,
                                                                'parallelism': None, 'template_config': {'envs': None, 'image':
                                                                'dptechnology/dpgen2:latest', 'retry_on_transient_error':
                                                                None, 'timeout': None, 'timeout_as_transient_error': False}})
dpgen2.entrypoint.submit_args.variant_explore()
dpgen2.entrypoint.submit_args.variant_fp()
dpgen2.entrypoint.submit_args.variant_train()
dpgen2.entrypoint.submit_args.vasp_args()
```



## dpngen2.entrypoint.watch module

```
dpngen2.entrypoint.watch.update_finished_steps(wf, finished_keys: Optional[List[str]] = None,
                                              download: Optional[bool] = False, watching_keys:
                                              Optional[List[str]] = None, prefix: Optional[str] =
                                              None)
```

```
dpngen2.entrypoint.watch.watch(workflow_id, wf_config: Optional[Dict] = {}, watching_keys: Optional[List]
                               = ['prep-run-train', 'prep-run-lmp', 'prep-run-fp', 'collect-data'], frequency:
                               Optional[float] = 600.0, download: Optional[bool] = False, prefix:
                               Optional[str] = None)
```

## dpngen2.exploration package

### Subpackages

### dpngen2.exploration.report package

### Submodules

### dpngen2.exploration.report.naive\_report module

```
class dpngen2.exploration.report.naive_report.NaiveExplorationReport(counter_f, counter_v)
    Bases: ExplorationReport
```

### Methods

<b>accurate_ratio</b>	
<b>calculate_ratio</b>	
<b>candidate_ratio</b>	
<b>failed_ratio</b>	
<b>ratio</b>	

```
accurate_ratio(tag=None) → float
```

```
static calculate_ratio(cc, ca, cf)
```

```
candidate_ratio(tag=None) → float
```

```
failed_ratio(tag=None) → float
```

```
ratio(quantity: str, item: str) → float
```

**dpgen2.exploration.report.report module****class** dpgen2.exploration.report.report.**ExplorationReport**Bases: [ABC](#)**Methods**

<b>accurate_ratio</b>	
<b>candidate_ratio</b>	
<b>failed_ratio</b>	

**abstract** **accurate\_ratio**(*tag=None*) → float**abstract** **candidate\_ratio**(*tag=None*) → float**abstract** **failed\_ratio**(*tag=None*) → float**dpgen2.exploration.report.trajs\_report module****class** dpgen2.exploration.report.trajs\_report.**TrajsExplorationReport**Bases: [ExplorationReport](#)**Methods**

<a href="#"><i>get_candidates</i></a> ([ <i>max_nframes</i> ])	Get candidates.
<a href="#"><i>record_traj</i></a> ( <i>id_f_accu</i> , <i>id_f_cand</i> , <i>id_f_fail</i> , ...)	Record one trajectory.

<b>accurate_ratio</b>	
<b>candidate_ratio</b>	
<b>clear</b>	
<b>failed_ratio</b>	

**accurate\_ratio**(*tag=None*)**candidate\_ratio**(*tag=None*)**clear**()**failed\_ratio**(*tag=None*)**get\_candidates**(*max\_nframes: Optional[int] = None*) → List[Tuple[int, int]]Get candidates. If number of candidates is larger than *max\_nframes*, then randomly pick *max\_nframes* frames from the candidates.**Parameters****max\_nframes** int

The maximal number of frames of candidates.

**Returns****cand\_frames** List[Tuple[int,int]]

Candidate frames. A list of tuples: [(traj\_idx, frame\_idx), ...]

Record one trajectory. inputs are the indexes of candidate, accurate and failed frames.

## Submodules

```
class dpgen2.exploration.scheduler.convergence_check_stage_scheduler.ConvergenceCheckStageScheduler(stage_scheduler.ExplorationRatioSelector):
    """Converge f-S-eclector, convolutional float = 0.9, max Op-tions = Non-fatal_c boot = True"""
```

## Methods

<b>complete</b>	
<b>reached_max_iteration</b>	

### 11.1. dpgen2 package

**converged()**

Tell if the stage is converged

**Returns**

**converged bool**

the convergence

**plan\_next\_iteration**(*report: Optional[ExplorationReport] = None, trajs: Optional[List[Path]] = None*)  
 → Tuple[bool, ExplorationTaskGroup, ConfSelector]

Make the plan for the next iteration of the stage.

It checks the report of the current and all historical iterations of the stage, and tells if the iterations are converged. If not converged, it will plan the next iteration for the stage.

**Parameters**

**hist\_reports: List[ExplorationReport]**

The historical exploration report of the stage. If this is the first iteration of the stage, this list is empty.

**report**

[ExplorationReport] The exploration report of this iteration.

**confs: List[Path]**

A list of configurations generated during the exploration. May be used to generate new configurations for the next iteration.

**Returns**

**stg\_complete: bool**

If the stage completed. Two cases may happen: 1. converged. 2. when not fatal\_at\_max, not converged but reached max number of iterations.

**task: ExplorationTaskGroup**

A *ExplorationTaskGroup* defining the exploration of the next iteration. Should be *None* if the stage is converged.

**conf\_selector: ConfSelector**

The configuration selector for the next iteration. Should be *None* if the stage is converged.

**reached\_max\_iteration()****dpngen2.exploration.scheduler.scheduler module****class dpngen2.exploration.scheduler.scheduler.ExplorationScheduler**

Bases: *object*

The exploration scheduler.

**Methods**

<i>add_stage_scheduler</i> (stage_scheduler)	Add stage scheduler.
<i>complete</i> ()	Tell if all stages are converged.
<i>get_convergence_ratio</i> ()	Get the accurate, candidate and failed ratios of the iterations
<i>get_iteration</i> ()	Get the index of the current iteration.
<i>get_stage</i> ()	Get the index of current stage.
<i>get_stage_of_iterations</i> ()	Get the stage index and the index in the stage of iterations.
<i>plan_next_iteration</i> ([report, trajs])	Make the plan for the next DPGEN iteration.

print_convergence	
-------------------	--

**add\_stage\_scheduler**(*stage\_scheduler*: [StageScheduler](#))

Add stage scheduler.

All added schedulers can be treated as a *list* (order matters). Only one stage is converged, the iteration goes to the next iteration.

**Parameters**

**stage\_scheduler**: [StageScheduler](#)  
The added stage scheduler

**complete**()

Tell if all stages are converged.

**get\_convergence\_ratio**()

Get the accurate, candidate and failed ratios of the iterations

**Returns**

**accu** [np.ndarray](#)  
The accurate ratio. length of array the same as # iterations.  
**cand** [np.ndarray](#)  
The candidate ratio. length of array the same as # iterations.  
**fail** [np.ndarray](#)  
The failed ration. length of array the same as # iterations.

**get\_iteration**()

Get the index of the current iteration.

Iteration index increase when *self.plan\_next\_iteration* returns valid *lmp\_task\_grp* and *conf\_selector* for the next iteration.

**get\_stage**()

Get the index of current stage.

Stage index increases when the previous stage converges. Usually called after *self.plan\_next\_iteration*.

**get\_stage\_of\_iterations**()

Get the stage index and the index in the stage of iterations.

**plan\_next\_iteration**(*report*: [Optional](#)[[ExplorationReport](#)] = *None*, *trajs*: [Optional](#)[[List](#)[[Path](#)]] = *None*)  
→ [Tuple](#)[[bool](#), [ExplorationTaskGroup](#), [ConfSelector](#)]

Make the plan for the next DPGEN iteration.

**Parameters**

**report**  
[[ExplorationReport](#)] The exploration report of this iteration.  
**confs**: [List](#)[[Path](#)]  
A list of configurations generated during the exploration. May be used to generate new configurations for the next iteration.

**Returns**

**complete**: [bool](#)  
If all the DPGEN stages complete.  
**task**: [ExplorationTaskGroup](#)  
A [ExplorationTaskGroup](#) defining the exploration of the next iteration. Should be *None* if converged.  
**conf\_selector**: [ConfSelector](#)  
The configuration selector for the next iteration. Should be *None* if converged.

```
print_convergence()
```

## dpgen2.exploration.scheduler.stage\_scheduler module

```
class dpgen2.exploration.scheduler.stage_scheduler.StageScheduler
```

Bases: [ABC](#)

The scheduler for an exploration stage.

### Methods

<a href="#"><code>converged()</code></a>	Tell if the stage is converged
<a href="#"><code>plan_next_iteration</code></a> (report, trajs)	Make the plan for the next iteration of the stage.

```
abstract converged()
```

Tell if the stage is converged

#### Returns

**converged bool**

the convergence

```
abstract plan_next_iteration(report: ExplorationReport, trajs: List[Path]) → Tuple[bool,
                                         ExplorationTaskGroup, ConfSelector]
```

Make the plan for the next iteration of the stage.

It checks the report of the current and all historical iterations of the stage, and tells if the iterations are converged. If not converged, it will plan the next iteration for the stage.

#### Parameters

**hist\_reports: List[ExplorationReport]**

The historical exploration report of the stage. If this is the first iteration of the stage, this list is empty.

**report**

[ExplorationReport] The exploration report of this iteration.

**confs: List[Path]**

A list of configurations generated during the exploration. May be used to generate new configurations for the next iteration.

#### Returns

**stg\_complete: bool**

If the stage completed. Two cases may happen: 1. converged. 2. when not fatal\_at\_max, not converged but reached max number of iterations.

**task: ExplorationTaskGroup**

A *ExplorationTaskGroup* defining the exploration of the next iteration. Should be *None* if the stage is converged.

**conf\_selector: ConfSelector**

The configuration selector for the next iteration. Should be *None* if the stage is converged.

## dpngen2.exploration.selector package

### Submodules

### dpngen2.exploration.selector.conf\_filter module

**class** dpngen2.exploration.selector.conf\_filter.**ConfFilter**

Bases: [ABC](#)

### Methods

---

<a href="#"><i>check</i></a> (coords, cell, atom_types, nopbc)	Check if the configuration is valid.
--	--------------------------------------

---

**abstract** [\*check\*](#)(coords: array, cell: array, atom\_types: array, nopbc: bool) → bool

Check if the configuration is valid.

#### Parameters

##### coords

[numpy.array] The coordinates, numpy array of shape natoms x 3

##### cell

[numpy.array] The cell tensor. numpy array of shape 3 x 3

##### atom\_types

[numpy.array] The atom types. numpy array of shape natoms

##### nopbc

[bool] If no periodic boundary condition.

#### Returns

##### valid

[bool] *True* if the configuration is a valid configuration, else *False*.

**class** dpngen2.exploration.selector.conf\_filter.**ConfFilters**

Bases: [object](#)

### Methods

<b>add</b>	
<b>check</b>	

**add**(conf\_filter: ConfFilter) → ConfFilters

**check**(conf: System) → bool

**dpngen2.exploration.selector.conf\_selector module****class** dpngen2.exploration.selector.conf\_selector.**ConfSelector**Bases: [ABC](#)

Select configurations from trajectory and model deviation files.

**Methods****select**

**abstract select**(trajs: *List[Path]*, model\_devis: *List[Path]*, traj\_fmt: *str* = 'deepmd/npz', type\_map: *Optional[List[str]]* = None) → *Tuple[List[Path], ExplorationReport]*

**dpngen2.exploration.selector.conf\_selector\_frame module**

**class** dpngen2.exploration.selector.conf\_selector\_frame.**ConfSelectorLammpsFrames**(trust\_level, max\_numb\_sel: *Optional[int]* = None, conf\_filters: *Optional[ConfFilters]* = None)

Bases: [ConfSelector](#)

Select frames from trajectories as confs.

Parameters: trust\_level: [TrustLevel](#)

The trust level

**conf\_filter:** [ConfFilters](#)

The configuration filter

**Methods**


---

*select*(trajs, model\_devis[, traj\_fmt, type\_map])      Select configurations

---

**record\_one\_traj****record\_one\_traj**(traj, model\_devi, traj\_fmt, type\_map) → None

**select**(trajs: *List[Path]*, model\_devis: *List[Path]*, traj\_fmt: *str* = 'lammps/dump', type\_map: *Optional[List[str]]* = None) → *Tuple[List[Path], ExplorationReport]*

Select configurations

**Parameters****trajs***[List[Path]]* A list of *Path* to trajectory files generated by LAMMPS



**model\_devis**

[List[Path]] A *list* of *Path* to model deviation files generated by LAMMPS. Format: each line has 7 numbers they are used as # frame\_id md\_v\_max md\_v\_min md\_v\_mean md\_f\_max md\_f\_min md\_f\_mean where *md* stands for model deviation, *v* for virial and *f* for force

**traj\_fmt**

[str] Format of the trajectory, by default it is the dump file of LAMMPS

**type\_map**

[List[str]] The *type\_map* of the systems

**Returns****confs**

[List[Path]] The selected configurations, stored in a folder in deepmd/npv format, can be parsed as dpdata.MultiSystems. The *list* only has one item.

**report**

[ExplorationReport] The exploration report recoding the status of the exploration.

**dpngen2.exploration.selector.trust\_level module**

```
class dpngen2.exploration.selector.trust_level.TrustLevel(level_f_lo, level_f_hi, level_v_lo=None,
                                                         level_v_hi=None)
```

Bases: `object`

**Attributes**

`level_f_hi`

`level_f_lo`

`level_v_hi`

`level_v_lo`

property `level_f_hi`

property `level_f_lo`

property `level_v_hi`

property `level_v_lo`

**dpngen2.exploration.task package****Subpackages****dpngen2.exploration.task.lmp package****Submodules****dpngen2.exploration.task.lmp.lmp\_input module**

```
dpngen2.exploration.task.lmp.lmp_input.make_lmp_input(conf_file: str, ensemble: str, graphs: List[str],
nsteps: int, dt: float, neidelay: int, trj_freq: int,
mass_map: List[float], temp: float, tau_t: float
= 0.1, pres: Optional[float] = None, tau_p:
float = 0.5, use_clusters: bool = False,
relative_f_epsilon: Optional[float] = None,
relative_v_epsilon: Optional[float] = None,
pka_e: Optional[float] = None, ele_temp_f:
Optional[float] = None, ele_temp_a:
Optional[float] = None, nopbc: bool = False,
max_seed: int = 1000000,
deepmd_version='2.0',
trj_seperate_files=True)
```

## Submodules

### dpngen2.exploration.task.npt\_task\_group module

**class** dpngen2.exploration.task.npt\_task\_group.NPTTaskGroup

Bases: *ExplorationTaskGroup*

#### Attributes

**task\_list**

Get the list of *ExplorationTask*

#### Methods

<code>add_group(group)</code>	Add another group to the group.
<code>add_task(task)</code>	Add one task to the group.
<code>count(value)</code>	
<code>index(value, [start, [stop]])</code>	Raises ValueError if the value is not present.
<code>make_task()</code>	Make the LAMMPS task group.
<code>set_conf(conf_list[, n_sample, random_sample])</code>	Set the configurations of exploration
<code>set_md(numb_models, mass_map, temps[, ...])</code>	Set MD parameters

clear

**make\_task()** → *ExplorationTaskGroup*

Make the LAMMPS task group.

#### Returns

**task\_grp:** *ExplorationTaskGroup*

The returned lammps task group. The number of tasks is  $nconf \cdot nT \cdot nP$ .  $nconf$  is set by  $n\_sample$  parameter of *set\_conf*.  $nT$  and  $nP$  are lengths of the *temps* and *press* parameters of *set\_md*.

**set\_conf**(conf\_list: List[str], n\_sample: Optional[int] = None, random\_sample: bool = False)

Set the configurations of exploration

#### Parameters

**conf\_list** str

A list of file contents

**n\_sample** int

Number of samples drawn from the conf list each time *make\_task* is called. If set to *None*, *n\_sample* is set to length of the *conf\_list*.

**random\_sample** bool

If true the confs are randomly sampled, otherwise are consecutively sampled from the *conf\_list*

**set\_md**(*numb\_models*, *mass\_map*, *temps*: *List*[float], *press*: *Optional*[*List*[float]] = *None*, *ens*: str = 'npt', *dt*: float = 0.001, *nsteps*: int = 1000, *trj\_freq*: int = 10, *tau\_t*: float = 0.1, *tau\_p*: float = 0.5, *pka\_e*: *Optional*[float] = *None*, *neidelay*: *Optional*[int] = *None*, *no\_pbc*: bool = *False*, *use\_clusters*: bool = *False*, *relative\_f\_epsilon*: *Optional*[float] = *None*, *relative\_v\_epsilon*: *Optional*[float] = *None*, *ele\_temp\_f*: *Optional*[float] = *None*, *ele\_temp\_a*: *Optional*[float] = *None*)

Set MD parameters

**dpngen2.exploration.task.stage module**

**class** dpngen2.exploration.task.stage.**ExplorationStage**

Bases: *object*

The exploration stage.

**Methods**

<i>add_task_group</i> (grp)	Add an exploration group
<i>clear</i> ()	Clear all exploration group.
<i>make_task</i> ()	Make the LAMMPS task group.

**add\_task\_group**(grp: *ExplorationTaskGroup*)

Add an exploration group

**Parameters**

**grp**: *ExplorationTaskGroup*

The added exploration task group

**clear**()

Clear all exploration group.

**make\_task**() → *ExplorationTaskGroup*

Make the LAMMPS task group.

**Returns**

**task\_grp**: *ExplorationTaskGroup*

The returned lammps task group. The number of tasks is equal to the summation of task groups defined by all the exploration groups added to the stage.

**dpgen2.exploration.task.task module**

**class** dpgen2.exploration.task.task.**ExplorationTask**

Bases: `object`

Define the files needed by an exploration task.

**Examples**

```
>>> # this example dumps all files needed by the task.
>>> files = exploration_task.files()
... for file_name, file_content in files.items():
...     with open(file_name, 'w') as fp:
...         fp.write(file_content)
```

**Methods**

<code>add_file(fname, fcont)</code>	Add file to the task
<code>files()</code>	Get all files for the task.

**add\_file**(*fname: str, fcont: str*)

Add file to the task

**Parameters**

**fname**

[str] The name of the file

**fcont**

[str] The content of the file.

**files()** → `Dict`

Get all files for the task.

**Returns**

**files**

[dict] The dict storing all files for the task. The file name is a key of the dict, and the file content is the corresponding value.

**class** dpgen2.exploration.task.task.**ExplorationTaskGroup**

Bases: `Sequence`

A group of exploration tasks. Implemented as a *list* of *ExplorationTask*.

**Attributes**

**task\_list**

Get the *list* of *ExplorationTask*

## Methods

<code>add_group(group)</code>	Add another group to the group.
<code>add_task(task)</code>	Add one task to the group.
<code>count(value)</code>	
<code>index(value, [start, [stop]])</code>	Raises ValueError if the value is not present.

clear	
-------	--

**add\_group**(*group*: [ExplorationTaskGroup](#))

Add another group to the group.

**add\_task**(*task*: [ExplorationTask](#))

Add one task to the group.

**clear**() → [None](#)

**property task\_list**: [List](#)[[ExplorationTask](#)]

Get the *list* of [ExplorationTask](#)

```
class dpngen2.exploration.task.task.FooTask(conf_name='conf.lmp', conf_cont="",
                                             inpu_name='in.lammps', inpu_cont="")
```

Bases: [ExplorationTask](#)

## Methods

<code>add_file(fname, fcont)</code>	Add file to the task
<code>files()</code>	Get all files for the task.

```
class dpngen2.exploration.task.task.FooTaskGroup(numb_task)
```

Bases: [ExplorationTaskGroup](#)

**Attributes**

[task\\_list](#)

Get the *list* of [ExplorationTask](#)

## Methods

<code>add_group(group)</code>	Add another group to the group.
<code>add_task(task)</code>	Add one task to the group.
<code>count(value)</code>	
<code>index(value, [start, [stop]])</code>	Raises ValueError if the value is not present.

clear	
-------	--

**property task\_list**

Get the *list* of [ExplorationTask](#)

## dpgen2.flow package

### Submodules

#### dpgen2.flow.dpgen\_loop module

```
class dpgen2.flow.dpgen_loop.ConcurrentLearning(name: str, block_op: OPTemplate, step_config: dict =
                                                {'continue_on_failed': False,
                                                 'continue_on_num_success': None,
                                                 'continue_on_success_ratio': None, 'executor': None,
                                                 'parallelism': None, 'template_config': {'envs': None,
                                                 'image': 'dptechnology/dpgen2:latest',
                                                 'retry_on_transient_error': None, 'timeout': None,
                                                 'timeout_as_transient_error': False}},
                                                upload_python_package: Optional[str] = None)
```

Bases: [Steps](#)

#### Attributes

**init\_keys**  
**input\_artifacts**  
**input\_parameters**  
**loop\_keys**  
**output\_artifacts**  
**output\_parameters**

#### Methods

---

add(step)	Add a step or a list of parallel steps to the steps
-----------	---

---

<b>convert_to_argo</b>	
<b>handle_key</b>	
<b>run</b>	

property **init\_keys**

property **input\_artifacts**

property **input\_parameters**

property **loop\_keys**

property **output\_artifacts**

property **output\_parameters**

```
class dpngen2.flow.dpngen_loop.ConcurrentLearningLoop(name: str, block_op: OPTemplate, step_config:
    dict = {'continue_on_failed': False,
            'continue_on_num_success': None,
            'continue_on_success_ratio': None, 'executor':
            None, 'parallelism': None, 'template_config':
            {'envs': None, 'image':
            'dptechnology/dpngen2:latest',
            'retry_on_transient_error': None, 'timeout':
            None, 'timeout_as_transient_error': False}},
            upload_python_package: Optional[str] = None)
```

Bases: [Steps](#)

#### Attributes

input\_artifacts  
input\_parameters  
keys  
output\_artifacts  
output\_parameters

#### Methods

add(step)	Add a step or a list of parallel steps to the steps
-----------	---

convert_to_argo	
handle_key	
run	

property input\_artifacts  
property input\_parameters  
property keys  
property output\_artifacts  
property output\_parameters

```
class dpngen2.flow.dpngen_loop.MakeBlockId(*args, **kwargs)
```

Bases: [OP](#)

#### Methods

execute(ip)	Run the OP
get_input_sign()	Get the signature of the inputs
get_output_sign()	Get the signature of the outputs

exec_sign_check	
function	
get_input_artifact_link	
get_input_artifact_storage_key	
get_output_artifact_link	
get_output_artifact_storage_key	

**execute**(*ip: OPIO*) → *OPIO*

Run the OP

**classmethod** **get\_input\_sign**()

Get the signature of the inputs

**classmethod** **get\_output\_sign**()

Get the signature of the outputs

**class** `dpgen2.flow.dpgen_loop.SchedulerWrapper(*args, **kwargs)`

Bases: *OP*

### Methods

<i>execute</i> ( <i>ip</i> )	Run the OP
<i>get_input_sign</i> ()	Get the signature of the inputs
<i>get_output_sign</i> ()	Get the signature of the outputs

exec_sign_check	
function	
get_input_artifact_link	
get_input_artifact_storage_key	
get_output_artifact_link	
get_output_artifact_storage_key	

**execute**(*ip: OPIO*) → *OPIO*

Run the OP

**classmethod** **get\_input\_sign**()

Get the signature of the inputs

**classmethod** **get\_output\_sign**()

Get the signature of the outputs



## dpngen2.fp package

### Submodules

#### dpngen2.fp.vasp module

```
class dpngen2.fp.vasp.VaspInputs(kspacing: Union[float, List[float]], kgamma: bool = True,
                                incar_template_name: Optional[str] = None, potcar_names:
                                Optional[Dict[str, str]] = None)
```

Bases: `object`

#### Attributes

`incar_template`  
`potcars`

#### Methods

<code>incar_from_file</code>	
<code>make_kpoints</code>	
<code>make_potcar</code>	
<code>potcars_from_file</code>	

`incar_from_file(fname: str)`

`property incar_template`

`make_kpoints(box: array) → str`

`make_potcar(atom_names) → str`

`property potcars`

`potcars_from_file(dict_fnames: Dict[str, str])`

`dpngen2.fp.vasp.make_kspacing_kpoints(box, kspacing, kgamma)`

## dpngen2.op package

### Submodules

#### dpngen2.op.collect\_data module

```
class dpngen2.op.collect_data.CollectData(*args, **kwargs)
```

Bases: `OP`

Collect labeled data and add to the iteration dataset.

After running FP tasks, the labeled data are scattered in task directories. This OP collect the labeled data in one data directory and add it to the iteration data. The data generated by this iteration will be place in `ip["name"]` subdirectory of the iteration data directory.

## Methods

<code>execute(ip)</code>	Execute the OP.
<code>get_input_sign()</code>	Get the signature of the inputs
<code>get_output_sign()</code>	Get the signature of the outputs

<code>exec_sign_check</code>	
<code>function</code>	
<code>get_input_artifact_link</code>	
<code>get_input_artifact_storage_key</code>	
<code>get_output_artifact_link</code>	
<code>get_output_artifact_storage_key</code>	

**execute**(*ip*: *OPIO*) → *OPIO*

Execute the OP. This OP collect data scattered in directories given by *ip*['*labeled\_data*'] in to one *dp-data.Multisystems* and store it in a directory named *name*. This directory is appended to the list *iter\_data*.

### Parameters

**ip**

[dict] Input dict with components:

- *name*: (*str*) The name of this iteration. The data generated by this iteration will be place in a sub-directory of *name*.
- *labeled\_data*: (*Artifact(List[Path])*) The paths of labeled data generated by FP tasks of the current iteration.
- *iter\_data*: (*Artifact(List[Path])*) The data paths previous iterations.

### Returns

Output dict with components:

- *iter\_data*: (*Artifact(List[Path])*) The data paths of previous and the current iteration data.

**classmethod** `get_input_sign()`

Get the signature of the inputs

**classmethod** `get_output_sign()`

Get the signature of the outputs

## dpngen2.op.md\_settings module

```
class dpngen2.op.md_settings.MDSettings(ens: str, dt: float, nsteps: int, trj_freq: int, temps:
Optional[List[float]] = None, press: Optional[List[float]] =
None, tau_t: float = 0.1, tau_p: float = 0.5, pka_e:
Optional[float] = None, neidelay: Optional[int] = None, no_pbc:
bool = False, use_clusters: bool = False, relative_epsilon:
Optional[float] = None, relative_v_epsilon: Optional[float] =
None, ele_temp_f: Optional[float] = None, ele_temp_a:
Optional[float] = None)
```

Bases: `object`

## Methods

to_str	
--------	--

**to\_str()** → str

## dpngen2.op.prep\_dp\_train module

**class** dpngen2.op.prep\_dp\_train.PrepDPTrain(\*args, \*\*kwargs)

Bases: OP

Prepares the working directories for DP training tasks.

A list of (*numb\_models*) working directories containing all files needed to start training tasks will be created. The paths of the directories will be returned as *op*[“task\_paths”]. The identities of the tasks are returned as *op*[“task\_names”].

## Methods

<a href="#"><i>execute</i>(ip)</a>	Execute the OP.
<a href="#"><i>get_input_sign</i>()</a>	Get the signature of the inputs
<a href="#"><i>get_output_sign</i>()</a>	Get the signature of the outputs

<b>exec_sign_check</b>	
<b>function</b>	
<b>get_input_artifact_link</b>	
<b>get_input_artifact_storage_key</b>	
<b>get_output_artifact_link</b>	
<b>get_output_artifact_storage_key</b>	

**execute**(ip: *OPIO*) → *OPIO*

Execute the OP.

### Parameters

**ip**

[dict] Input dict with components:

- *template\_script*: (*str* or *List[str]*) A template of the training script. Can be a *str* or *List[str]*. In the case of *str*, all training tasks share the same training input template, the only difference is the random number used to initialize the network parameters. In the case of *List[str]*, one training task uses one template from the list. The random numbers used to initialize the network parameters are different. The length of the list should be the same as *numb\_models*.
- *numb\_models*: (*int*) Number of DP models to train.

### Returns

**op**

[dict] Output dict with components:

- *task\_names*: (*List[str]*) The name of tasks. Will be used as the identities of the tasks. The names of different tasks are different.

- *task\_paths*: (*Artifact(List[Path])*) The prepared working paths of the tasks. The order of the Paths should be consistent with *op["task\_names"]*

**classmethod** `get_input_sign()`

Get the signature of the inputs

**classmethod** `get_output_sign()`

Get the signature of the outputs

## dpngen2.op.prep\_lmp module

`dpngen2.op.prep_lmp.PrepExplorationTaskGroup`

alias of *PrepLmp*

**class** `dpngen2.op.prep_lmp.PrepLmp(*args, **kwargs)`

Bases: *OP*

Prepare the working directories for LAMMPS tasks.

A list of working directories (defined by *ip["task"]*) containing all files needed to start LAMMPS tasks will be created. The paths of the directories will be returned as *op["task\_paths"]*. The identities of the tasks are returned as *op["task\_names"]*.

## Methods

<code>execute(ip)</code>	Execute the OP.
<code>get_input_sign()</code>	Get the signature of the inputs
<code>get_output_sign()</code>	Get the signature of the outputs

<b>exec_sign_check</b>	
<b>function</b>	
<b>get_input_artifact_link</b>	
<b>get_input_artifact_storage_key</b>	
<b>get_output_artifact_link</b>	
<b>get_output_artifact_storage_key</b>	

**execute**(*ip*: *OPIO*) → *OPIO*

Execute the OP.

### Parameters

**ip**

[dict] Input dict with components: - *lmp\_task\_grp* : (*Artifact(Path)*) Can be pickle loaded as a ExplorationTaskGroup. Definitions for LAMMPS tasks

### Returns

**op**

[dict] Output dict with components:

- *task\_names*: (*List[str]*) The name of tasks. Will be used as the identities of the tasks. The names of different tasks are different.
- *task\_paths*: (*Artifact(List[Path])*) The prepared working paths of the tasks. Contains all input files needed to start the LAMMPS simulation. The order of the Paths should be consistent with *op["task\_names"]*

```
classmethod get_input_sign()
    Get the signature of the inputs

classmethod get_output_sign()
    Get the signature of the outputs
```

## dpngen2.op.prep\_vasp module

```
class dpngen2.op.prep_vasp.PrepVasp(*args, **kwargs)
```

Bases: `OP`

Prepares the working directories for VASP tasks.

A list of (same length as `ip["confs"]`) working directories containing all files needed to start VASP tasks will be created. The paths of the directories will be returned as `op["task_paths"]`. The identities of the tasks are returned as `op["task_names"]`.

## Methods

<code>execute(ip)</code>	Execute the OP.
<code>get_input_sign()</code>	Get the signature of the inputs
<code>get_output_sign()</code>	Get the signature of the outputs

<code>exec_sign_check</code>	
<code>function</code>	
<code>get_input_artifact_link</code>	
<code>get_input_artifact_storage_key</code>	
<code>get_output_artifact_link</code>	
<code>get_output_artifact_storage_key</code>	

**execute**(*ip*: *OPIO*) → *OPIO*

Execute the OP.

### Parameters

**ip**

[dict] Input dict with components:

- *inputs* : (*VaspInputs*) Definitions for the VASP inputs
- *confs* : (*Artifact(List[Path])*) Configurations for the VASP tasks. Stored in folders as deepmd/npz format. Can be parsed as `dpdata.MultiSystems`.

### Returns

**op**

[dict] Output dict with components:

- *task\_names*: (*List[str]*) The name of tasks. Will be used as the identities of the tasks. The names of different tasks are different.
- *task\_paths*: (*Artifact(List[Path])*) The prepared working paths of the tasks. Contains all input files needed to start the VASP. The order for the Paths should be consistent with `op["task_names"]`

```
classmethod get_input_sign()
    Get the signature of the inputs

classmethod get_output_sign()
    Get the signature of the outputs
```

## dpgen2.op.run\_dp\_train module

```
class dpgen2.op.run_dp_train.RunDPTrain(*args, **kwargs)
```

Bases: `OP`

Execute a DP training task. Train and freeze a DP model.

A working directory named *task\_name* is created. All input files are copied or symbol linked to directory *task\_name*. The DeePMD-kit training and freezing commands are executed from directory *task\_name*.

## Methods

<code>execute(ip)</code>	Execute the OP.
<code>get_input_sign()</code>	Get the signature of the inputs
<code>get_output_sign()</code>	Get the signature of the outputs

<code>decide_init_model</code>	
<code>exec_sign_check</code>	
<code>function</code>	
<code>get_input_artifact_link</code>	
<code>get_input_artifact_storage_key</code>	
<code>get_output_artifact_link</code>	
<code>get_output_artifact_storage_key</code>	
<code>normalize_config</code>	
<code>training_args</code>	
<code>write_data_to_input_script</code>	
<code>write_other_to_input_script</code>	

```
static decide_init_model(config, init_model, init_data, iter_data)
```

```
execute(ip: OPIO) → OPIO
```

Execute the OP.

### Parameters

**ip**

[dict] Input dict with components:

- *config*: (*dict*) The config of training task. Check *RunDPTrain.training\_args* for definitions.
- *task\_name*: (*str*) The name of training task.
- *task\_path*: (*Artifact(Path)*) The path that contains all input files prepared by *PrepDPTrain*.
- *init\_model*: (*Artifact(Path)*) A frozen model to initialize the training.
- *init\_data*: (*Artifact(List[Path])*) Initial training data.

- *iter\_data*: (*Artifact(List[Path])*) Training data generated in the DPGEN iterations.

#### Returns

Output dict with components:

- *script*: (*Artifact(Path)*) The training script.
- *model*: (*Artifact(Path)*) The trained frozen model.
- *lcurve*: (*Artifact(Path)*) The learning curve file.
- *log*: (*Artifact(Path)*) The log file of training.

**classmethod** `get_input_sign()`

Get the signature of the inputs

**classmethod** `get_output_sign()`

Get the signature of the outputs

**static** `normalize_config(data={})`

**static** `training_args()`

**static** `write_data_to_input_script(idict: dict, init_data: List[Path], iter_data: List[Path], auto_prob_str: str = 'prob_sys_size', major_version: str = '1')`

**static** `write_other_to_input_script(idict, config, do_init_model, major_version: str = '1')`

`dpngen2.op.run_dp_train.config_args()`

### dpngen2.op.run\_lmp module

**class** `dpngen2.op.run_lmp.RunLmp(*args, **kwargs)`

Bases: `OP`

Execute a LAMMPS task.

A working directory named *task\_name* is created. All input files are copied or symbol linked to directory *task\_name*. The LAMMPS command is executed from directory *task\_name*. The trajectory and the model deviation will be stored in files *op["traj"]* and *op["model\_devi"]*, respectively.

#### Methods

<code>execute(ip)</code>	Execute the OP.
<code>get_input_sign()</code>	Get the signature of the inputs
<code>get_output_sign()</code>	Get the signature of the outputs

<code>exec_sign_check</code>	
<code>function</code>	
<code>get_input_artifact_link</code>	
<code>get_input_artifact_storage_key</code>	
<code>get_output_artifact_link</code>	
<code>get_output_artifact_storage_key</code>	
<code>lmp_args</code>	
<code>normalize_config</code>	

**execute**(*ip*: *OPIO*) → *OPIO*

Execute the OP.

**Parameters**

**ip**

[dict] Input dict with components:

- *config*: (*dict*) The config of *lmp* task. Check *RunLmp.lmp\_args* for definitions.
- *task\_name*: (*str*) The name of the task.
- *task\_path*: (*Artifact(Path)*) The path that contains all input files prepared by *PrepLmp*.
- *models*: (*Artifact(List[Path])*) The frozen model to estimate the model deviation. The first model will be used to drive molecular dynamics simulation.

**Returns**

**Output dict with components:**

- *log*: (*Artifact(Path)*) The log file of LAMMPS.
- *traj*: (*Artifact(Path)*) The output trajectory.
- *model\_devi*: (*Artifact(Path)*) The model deviation. The order of recorded model deviations should be consistent with the order of frames in *traj*.

**classmethod** *get\_input\_sign*()

Get the signature of the inputs

**classmethod** *get\_output\_sign*()

Get the signature of the outputs

**static** *lmp\_args*()

**static** *normalize\_config*(*data*={})

*dpngen2.op.run\_lmp.config\_args*()

## **dpngen2.op.run\_vasp module**

**class** *dpngen2.op.run\_vasp.RunVasp*(\*args, \*\*kwargs)

Bases: *OP*

Execute a VASP task.

A working directory named *task\_name* is created. All input files are copied or symbol linked to directory *task\_name*. The VASP command is executed from directory *task\_name*. The *op*["*labeled\_data*"] in "*deepmd/npz*" format (HF5 in the future) provided by *dpdata* will be created.



## Methods

<code>execute(ip)</code>	Execute the OP.
<code>get_input_sign()</code>	Get the signature of the inputs
<code>get_output_sign()</code>	Get the signature of the outputs

<code>exec_sign_check</code>	
<code>function</code>	
<code>get_input_artifact_link</code>	
<code>get_input_artifact_storage_key</code>	
<code>get_output_artifact_link</code>	
<code>get_output_artifact_storage_key</code>	
<code>normalize_config</code>	
<code>vasp_args</code>	

**execute**(*ip*: *OPIO*) → *OPIO*

Execute the OP.

### Parameters

**ip**

[dict] Input dict with components:

- *config*: (*dict*) The config of vasp task. Check *RunVasp.vasp\_args* for definitions.
- *task\_name*: (*str*) The name of task.
- *task\_path*: (*Artifact(Path)*) The path that contains all input files prepared by *PrepVasp*.

### Returns

Output dict with components:

- *log*: (*Artifact(Path)*) The log file of VASP.
- *labeled\_data*: (*Artifact(Path)*) The path to the labeled data in “*deepmd/npz*” format provided by *dpdata*.

**classmethod** `get_input_sign()`

Get the signature of the inputs

**classmethod** `get_output_sign()`

Get the signature of the outputs

**static** `normalize_config(data={})`

**static** `vasp_args()`

`dpngen2.op.run_vasp.config_args()`

**dpgen2.op.select\_confs module**

**class** dpgen2.op.select\_confs.**SelectConfs**(\*args, \*\*kwargs)

Bases: `OP`

Select configurations from exploration trajectories for labeling.

**Methods**

<code>execute(ip)</code>	Execute the OP.
<code>get_input_sign()</code>	Get the signature of the inputs
<code>get_output_sign()</code>	Get the signature of the outputs

<code>exec_sign_check</code>	
<code>function</code>	
<code>get_input_artifact_link</code>	
<code>get_input_artifact_storage_key</code>	
<code>get_output_artifact_link</code>	
<code>get_output_artifact_storage_key</code>	

**execute**(*ip*: *OPIO*) → *OPIO*

Execute the OP.

**Parameters**

**ip**

[dict] Input dict with components:

- *conf\_selector*: (*ConfSelector*) Configuration selector.
- *traj\_fmt*: (*str*) The format of trajectory.
- *type\_map*: (*List[str]*) The type map.
- *trajs*: (*Artifact(List[Path])*) The trajectories generated in the exploration.
- *model\_devis*: (*Artifact(List[Path])*) The file storing the model deviation of the trajectory. The order of model deviation storage is consistent with that of the trajectories. The order of frames of one model deviation storage is also consistent with tat of the corresponding trajectory.

**Returns**

**Output dict with components:**

- *report*: (*ExplorationReport*) The report on the exploration.
- *conf*: (*Artifact(List[Path])*) The selected configurations.

**classmethod** `get_input_sign()`

Get the signature of the inputs

**classmethod** `get_output_sign()`

Get the signature of the outputs

## dpngen2.superop package

### Submodules

## dpngen2.superop.block module

```
class dpngen2.superop.block.ConcurrentLearningBlock(name: str, prep_run_dp_train_op: OPTemplate,
                                                    prep_run_lmp_op: OPTemplate, select_confs_op:
                                                    OP, prep_run_fp_op: OPTemplate,
                                                    collect_data_op: OP, select_confs_config: dict =
                                                    {'continue_on_failed': False,
                                                    'continue_on_num_success': None,
                                                    'continue_on_success_ratio': None, 'executor':
                                                    None, 'parallelism': None, 'template_config':
                                                    {'envs': None, 'image':
                                                    'dptechology/dpgen2:latest',
                                                    'retry_on_transient_error': None, 'timeout': None,
                                                    'timeout_as_transient_error': False}},
                                                    collect_data_config: dict = {'continue_on_failed':
                                                    False, 'continue_on_num_success': None,
                                                    'continue_on_success_ratio': None, 'executor':
                                                    None, 'parallelism': None, 'template_config':
                                                    {'envs': None, 'image':
                                                    'dptechology/dpgen2:latest',
                                                    'retry_on_transient_error': None, 'timeout': None,
                                                    'timeout_as_transient_error': False}},
                                                    upload_python_package: Optional[str] = None)
```

Bases: [Steps](#)

#### Attributes

**input\_artifacts**  
**input\_parameters**  
**keys**  
**output\_artifacts**  
**output\_parameters**

#### Methods

---

add(step)	Add a step or a list of parallel steps to the steps
-----------	---

---

<b>convert_to_argo</b>	
<b>handle_key</b>	
<b>run</b>	

**property** input\_artifacts  
**property** input\_parameters  
**property** keys  
**property** output\_artifacts

property output\_parameters

## dpgen2.superop.prep\_run\_dp\_train module

```
class dpgen2.superop.prep_run_dp_train.PrepRunDPTrain(name: str, prep_train_op: OP, run_train_op:
    OP, prep_config: dict =
    {'continue_on_failed': False,
    'continue_on_num_success': None,
    'continue_on_success_ratio': None,
    'executor': None, 'parallelism': None,
    'template_config': {'envs': None, 'image':
    'dptechnology/dpgen2:latest',
    'retry_on_transient_error': None, 'timeout':
    None, 'timeout_as_transient_error': False}},
    run_config: dict = {'continue_on_failed':
    False, 'continue_on_num_success': None,
    'continue_on_success_ratio': None,
    'executor': None, 'parallelism': None,
    'template_config': {'envs': None, 'image':
    'dptechnology/dpgen2:latest',
    'retry_on_transient_error': None, 'timeout':
    None, 'timeout_as_transient_error': False}},
    upload_python_package: Optional[str] =
    None)
```

Bases: [Steps](#)

### Attributes

input\_artifacts  
input\_parameters  
keys  
output\_artifacts  
output\_parameters

### Methods

add(step)	Add a step or a list of parallel steps to the steps
-----------	---

convert_to_argo	
handle_key	
run	

property input\_artifacts  
property input\_parameters  
property keys  
property output\_artifacts  
property output\_parameters

**dpngen2.superop.prep\_run\_fp module**

```

class dpngen2.superop.prep_run_fp.PrepRunFp(name: str, prep_op: OP, run_op: OP, prep_config: dict =
    {'continue_on_failed': False, 'continue_on_num_success':
    None, 'continue_on_success_ratio': None, 'executor': None,
    'parallelism': None, 'template_config': {'envs': None,
    'image': 'dptechnology/dpgen2:latest',
    'retry_on_transient_error': None, 'timeout': None,
    'timeout_as_transient_error': False}}, run_config: dict =
    {'continue_on_failed': False, 'continue_on_num_success':
    None, 'continue_on_success_ratio': None, 'executor': None,
    'parallelism': None, 'template_config': {'envs': None,
    'image': 'dptechnology/dpgen2:latest',
    'retry_on_transient_error': None, 'timeout': None,
    'timeout_as_transient_error': False}},
    upload_python_package: Optional[str] = None)

```

Bases: [Steps](#)**Attributes**

[input\\_artifacts](#)  
[input\\_parameters](#)  
[keys](#)  
[output\\_artifacts](#)  
[output\\_parameters](#)

**Methods**

<a href="#">add(step)</a>	Add a step or a list of parallel steps to the steps
---------------------------	---

<a href="#">convert_to_argo</a>	
<a href="#">handle_key</a>	
<a href="#">run</a>	

[property input\\_artifacts](#)  
[property input\\_parameters](#)  
[property keys](#)  
[property output\\_artifacts](#)  
[property output\\_parameters](#)

**dpgen2.superop.prep\_run\_lmp module**

```
class dpgen2.superop.prep_run_lmp.PrepRunLmp(name: str, prep_op: OP, run_op: OP, prep_config: dict =
    {'continue_on_failed': False, 'continue_on_num_success':
    None, 'continue_on_success_ratio': None, 'executor':
    None, 'parallelism': None, 'template_config': {'envs':
    None, 'image': 'dptechnology/dpgen2:latest',
    'retry_on_transient_error': None, 'timeout': None,
    'timeout_as_transient_error': False}}, run_config: dict =
    {'continue_on_failed': False, 'continue_on_num_success':
    None, 'continue_on_success_ratio': None, 'executor':
    None, 'parallelism': None, 'template_config': {'envs':
    None, 'image': 'dptechnology/dpgen2:latest',
    'retry_on_transient_error': None, 'timeout': None,
    'timeout_as_transient_error': False}},
    upload_python_package: Optional[str] = None)
```

Bases: Steps

**Attributes**

input\_artifacts  
input\_parameters  
keys  
output\_artifacts  
output\_parameters

**Methods**

add(step)	Add a step or a list of parallel steps to the steps
-----------	---

convert_to_argo	
handle_key	
run	

property input\_artifacts  
property input\_parameters  
property keys  
property output\_artifacts  
property output\_parameters

## dpngen2.utils package

### Submodules

### dpngen2.utils.alloy\_conf module

**class** dpngen2.utils.alloy\_conf.**AlloyConf**(*lattice: Union[System, Tuple[str, float]]*, *type\_map: List[str]*, *replicate: Optional[Union[List[int], Tuple[int], int]] = None*)

Bases: `object`

#### Parameters

**lattice** `Union[dpdata.System, Tuple[str, float]]`

Lattice of the alloy confs. can be *dpdata.System*: lattice in *dpdata.System Tuple[str, float]*: pair of lattice type and lattice constant. lattice type can be “bcc”, “fcc”, “hcp”, “sc” or “diamond”

**replicate** `Union[List[int], Tuple[int], int]`

replicate of the lattice

**type\_map** `List[str]`

The type map

#### Methods

---

*generate\_file\_content*(numb\_confs[, ...])

#### Parameters

---

*generate\_systems*(numb\_confs[, ...])

#### Parameters

---

**generate\_file\_content**(*numb\_confs*, *concentration: Optional[Union[List[List[float]], List[float]]] = None*, *cell\_pert\_frac: float = 0.0*, *atom\_pert\_dist: float = 0.0*, *fmt: str = 'lammps/lmp'*)  $\rightarrow$  `List[str]`

#### Parameters

**numb\_confs** `int`

Number of configurations to generate

**concentration** `List[List[float]]` or `List[float]` or `None`

If *List[float]*, the concentrations of each element. The length of the list should be the same as the *type\_map*. If *List[List[float]]*, a list of concentrations (*List[float]*) is randomly picked from the List. If *None*, the elements are assumed to be of equal concentration.

**cell\_pert\_frac** `float`

fraction of cell perturbation

**atom\_pert\_dist** `float`

the atom perturbation distance (unit angstrom).

**fmt** `str`

the format of the returned conf strings. Should be one of the formats supported by *dpdata*

#### Returns

**conf\_list** `List[str]`

A list of file content of configurations.

**generate\_systems**(*numb\_confs*, *concentration*: *Optional[Union[List[List[float]], List[float]]] = None*,  
*cell\_pert\_frac*: *float = 0.0*, *atom\_pert\_dist*: *float = 0.0*) → *List[str]*

#### Parameters

**numb\_confs** *int*

Number of configurations to generate

**concentration** *List[List[float]] or List[float] or None*

If *List[float]*, the concentrations of each element. The length of the list should be the same as the *type\_map*. If *List[List[float]]*, a list of concentrations (*List[float]*) is randomly picked from the List. If *None*, the elements are assumed to be of equal concentration.

**cell\_pert\_frac** *float*

fraction of cell perturbation

**atom\_pert\_dist** *float*

the atom perturbation distance (unit angstrom).

#### Returns

**conf\_list** *List[dpdata.System]*

A list of generated confs in *dpdata.System*.

`dpngen2.utils.alloy_conf.gen_doc(*, make_anchor=True, make_link=True, **kwargs)`

`dpngen2.utils.alloy_conf.generate_alloy_conf_args()`

`dpngen2.utils.alloy_conf.generate_alloy_conf_file_content`(*lattice*: *Union[System, Tuple[str, float]]*,  
*type\_map*: *List[str]*, *numb\_confs*,  
*replicate*: *Optional[Union[List[int], Tuple[int], int]] = None*, *concentration*:  
*Optional[Union[List[List[float]], List[float]]] = None*, *cell\_pert\_frac*: *float*  
*= 0.0*, *atom\_pert\_dist*: *float = 0.0*, *fmt*: *str*  
*= 'lammps/lmp'*)

`dpngen2.utils.alloy_conf.normalize(data)`

## dpngen2.utils.chdir module

`dpngen2.utils.chdir.chdir(path_key: str)`

Returns a decorator that can change the current working path.

#### Parameters

**path\_key**

[str] key to OPIO

## Examples

```
>>> class SomeOP(OP):
...     @chdir("path")
...     def execute(self, ip: OPIO):
...         do_something()
```

`dpngen2.utils.chdir.set_directory(path: Path)`

Sets the current working path within the context.

#### Parameters



**path**  
 [Path] The path to the cwd  
**Yields**  
 None

### Examples

```
>>> with set_directory("some_path"):
...     do_something()
```

### dpngen2.utils.dflow\_config module

`dpngen2.utils.dflow_config.dflow_config(config_data)`

set the dflow config by *config\_data*

the keys starting with “**s3\_**” will be treated as s3\_config keys, other keys are treated as config keys.

`dpngen2.utils.dflow_config.dflow_config_lower(dflow_config)`

`dpngen2.utils.dflow_config.dflow_s3_config(config_data)`

set the s3 config by *config\_data*

`dpngen2.utils.dflow_config.dflow_s3_config_lower(dflow_s3_config_data)`

`dpngen2.utils.dflow_config.workflow_config_from_dict(wf_config)`

### dpngen2.utils.dflow\_query module

`dpngen2.utils.dflow_query.find_slice_ranges(keys: List[str], sliced_subkey: str)`

find range of sliced OPs that matches the pattern ‘iter-[0-9]\*-*{sliced\_subkey}*’-[0-9]\*’

`dpngen2.utils.dflow_query.get_iteration(key: str)`

`dpngen2.utils.dflow_query.get_last_iteration(keys: List[str])`

get the index of the last iteration from a list of step keys.

`dpngen2.utils.dflow_query.get_last_scheduler(wf: Any, keys: List[str])`

get the output Scheduler of the last successful iteration

`dpngen2.utils.dflow_query.get_subkey(key: str, idx: Optional[int] = -1)`

`dpngen2.utils.dflow_query.matched_step_key(all_keys: List[str], step_keys: Optional[List[str]] = None)`

returns the keys in *all\_keys* that matches any of the *step\_keys*

`dpngen2.utils.dflow_query.print_keys_in_nice_format(keys: List[str], sliced_subkey: List[str],  
 idx_fmt_len: int = 8)`

`dpngen2.utils.dflow_query.sort_slice_ops(keys: List[str], sliced_subkey: List[str])`

sort the keys of the sliced ops. the keys of the sliced ops contains *sliced\_subkey*

### dpgen2.utils.download\_dpgen2\_artifacts module

**class** dpgen2.utils.download\_dpgen2\_artifacts.**DownloadDefinition**

Bases: `object`

#### Methods

<b>add_def</b>	
<b>add_input</b>	
<b>add_output</b>	

**add\_def**(*tdict*, *key*, *suffix=None*)

**add\_input**(*input\_key*, *suffix=None*)

**add\_output**(*output\_key*, *suffix=None*)

**dpgen2.utils.download\_dpgen2\_artifacts.download\_dpgen2\_artifacts**(*wf*, *key*, *prefix=None*)

download the artifacts of a step. the key should be of format 'iter-xxxxxx-subkey-of-step-xxxxxx' the input and output artifacts will be downloaded to prefix/iter-xxxxxx/key-of-step/inputs/ and prefix/iter-xxxxxx/key-of-step/outputs/

the downloaded input and output artifacts of steps are defined by *op\_download\_setting*

### dpgen2.utils.obj\_artifact module

**dpgen2.utils.obj\_artifact.dump\_object\_to\_file**(*obj*, *fname*)

pickle dump object to a file

**dpgen2.utils.obj\_artifact.load\_object\_from\_file**(*fname*)

pickle load object from a file

### dpgen2.utils.run\_command module

**dpgen2.utils.run\_command.run\_command**(*cmd*, *shell=None*)

### dpgen2.utils.step\_config module

**dpgen2.utils.step\_config.gen\_doc**(*\**, *make\_anchor=True*, *make\_link=True*, *\*\*kwargs*)

**dpgen2.utils.step\_config.init\_executor**(*executor\_dict*)

**dpgen2.utils.step\_config.lebesgue\_executor\_args**()

**dpgen2.utils.step\_config.lebesgue\_extra\_args**()

**dpgen2.utils.step\_config.normalize**(*data*)

**dpgen2.utils.step\_config.step\_conf\_args**()

```
dpngen2.utils.step_config.template_conf_args()
```

```
dpngen2.utils.step_config.variant_executor()
```

### dpngen2.utils.unit\_cells module

```
class dpngen2.utils.unit_cells.BCC
```

Bases: `object`

#### Methods

<code>gen_box</code>	
<code>numb_atoms</code>	
<code>poscar_unit</code>	

```
gen_box()
```

```
numb_atoms()
```

```
poscar_unit(latt)
```

```
class dpngen2.utils.unit_cells.DIAMOND
```

Bases: `object`

#### Methods

<code>gen_box</code>	
<code>numb_atoms</code>	
<code>poscar_unit</code>	

```
gen_box()
```

```
numb_atoms()
```

```
poscar_unit(latt)
```

```
class dpngen2.utils.unit_cells.FCC
```

Bases: `object`

#### Methods

<code>gen_box</code>	
<code>numb_atoms</code>	
<code>poscar_unit</code>	

```
gen_box()
```

```
numb_atoms()
```

**poscar\_unit**(*latt*)

**class** dpgen2.utils.unit\_cells.HCP

Bases: [object](#)

### Methods

<b>gen_box</b>	
<b>numb_atoms</b>	
<b>poscar_unit</b>	

**gen\_box**()

**numb\_atoms**()

**poscar\_unit**(*latt*)

**class** dpgen2.utils.unit\_cells.SC

Bases: [object](#)

### Methods

<b>gen_box</b>	
<b>numb_atoms</b>	
<b>poscar_unit</b>	

**gen\_box**()

**numb\_atoms**()

**poscar\_unit**(*latt*)

dpgen2.utils.unit\_cells.**generate\_unit\_cell**(*crystal*: [str](#), *latt*: [float](#) = 1.0) → [System](#)

## 11.1.2 Submodules

### 11.1.3 dpgen2.constants module

- [genindex](#)
- [modindex](#)
- [search](#)

## PYTHON MODULE INDEX

### d

dpngen2, 73  
dpngen2.constants, 112  
dpngen2.entrypoint, 73  
dpngen2.entrypoint.download, 73  
dpngen2.entrypoint.main, 73  
dpngen2.entrypoint.showkey, 74  
dpngen2.entrypoint.status, 74  
dpngen2.entrypoint.submit, 74  
dpngen2.entrypoint.submit\_args, 76  
dpngen2.entrypoint.watch, 77  
dpngen2.exploration, 77  
dpngen2.exploration.report, 77  
dpngen2.exploration.report.naive\_report, 77  
dpngen2.exploration.report.report, 78  
dpngen2.exploration.report.trajs\_report, 78  
dpngen2.exploration.scheduler, 79  
dpngen2.exploration.scheduler.convergence\_check\_stage\_scheduler, 79  
dpngen2.exploration.scheduler.scheduler, 80  
dpngen2.exploration.scheduler.stage\_scheduler, 82  
dpngen2.exploration.selector, 83  
dpngen2.exploration.selector.conf\_filter, 83  
dpngen2.exploration.selector.conf\_selector, 84  
dpngen2.exploration.selector.conf\_selector\_frame, 84  
dpngen2.exploration.selector.trust\_level, 85  
dpngen2.exploration.task, 85  
dpngen2.exploration.task.lmp, 85  
dpngen2.exploration.task.lmp.lmp\_input, 85  
dpngen2.exploration.task.npt\_task\_group, 86  
dpngen2.exploration.task.stage, 87  
dpngen2.exploration.task.task, 88  
dpngen2.flow, 90  
dpngen2.flow.dpngen\_loop, 90  
dpngen2.fp, 93  
dpngen2.fp.vasp, 93  
dpngen2.op, 93  
dpngen2.op.collect\_data, 93  
dpngen2.op.md\_settings, 94  
dpngen2.op.prep\_dp\_train, 95  
dpngen2.op.prep\_lmp, 96  
dpngen2.op.prep\_vasp, 97  
dpngen2.op.run\_dp\_train, 98  
dpngen2.op.run\_lmp, 99  
dpngen2.op.run\_vasp, 100  
dpngen2.op.select\_confs, 102  
dpngen2.superop, 103  
dpngen2.superop.block, 103  
dpngen2.superop.prep\_run\_dp\_train, 104  
dpngen2.superop.prep\_run\_fp, 105  
dpngen2.superop.prep\_run\_lmp, 106  
dpngen2.utils, 107  
dpngen2.utils.alloy\_conf, 107  
dpngen2.utils.chdir, 108  
dpngen2.utils.dflow\_config, 109  
dpngen2.utils.dflow\_query, 109  
dpngen2.utils.download\_dpngen2\_artifacts, 110  
dpngen2.utils.ghi\_artifact, 110  
dpngen2.utils.run\_command, 110  
dpngen2.utils.step\_config, 110  
dpngen2.utils.unit\_cells, 111



## INDEX

### A

`accurate_ratio()` (dp- `candidate_ratio()` (dp- `gen2.exploration.report.naive_report.NaiveExplorationReport` static method), 77  
`gen2.exploration.report.naive_report.NaiveExplorationReport` method), 77  
`accurate_ratio()` (dp- `candidate_ratio()` (dp- `gen2.exploration.report.report.ExplorationReport` (dp- `gen2.exploration.report.report.ExplorationReport` method), 78  
`method`), 78  
`accurate_ratio()` (dp- `candidate_ratio()` (dp- `gen2.exploration.report.trajs_report.TrajsExplorationReport` (dp- `gen2.exploration.report.trajs_report.TrajsExplorationReport` method), 78  
`method`), 78  
`add()` (dp- `gen2.exploration.selector.conf_filter.ConfFilters` `cell_pert_frac (Argument)` method), 83  
`cell_pert_frac:` `cell_pert_frac:` `cell_pert_frac (Argument)`, 53  
`add_def()` (dp- `gen2.utils.download_dp- gen2_artifacts.DownloadDefinition method), 110  
method), 110  
add_file() (dp- gen2.exploration.task.task.ExplorationTask chdir() (in module dp- gen2.utils.chdir()), 108  
method), 88  
check() (dp- gen2.exploration.selector.conf_filter.ConfFilter method), 83  
add_group() (dp- gen2.exploration.task.task.ExplorationTaskGroup check() (dp- gen2.exploration.selector.conf_filter.ConfFilters method), 83  
method), 89  
add_input() (dp- gen2.utils.download_dp- gen2_artifacts.DownloadDefinition method), 110  
method), 110  
add_output() (dp- gen2.utils.download_dp- gen2_artifacts.DownloadDefinition method), 110  
method), 110  
add_stage_scheduler() (dp- gen2.exploration.scheduler.scheduler.ExplorationScheduler clear() (dp- gen2.exploration.task.stage.ExplorationStage method), 81  
method), 81  
add_task() (dp- gen2.exploration.task.task.ExplorationTaskGroup clear() (dp- gen2.exploration.task.task.ExplorationTaskGroup method), 89  
method), 89  
add_task_group() (dp- gen2.exploration.task.stage.ExplorationStage collect_data_config: step_configs/collect_data_config (Argument), 40  
method), 87  
AlloyConf (class in dp- gen2.utils.alloy_conf), 107  
atom_pert_dist (Argument) CollectData (class in dp- gen2.op.collect_data), 93  
atom_pert_dist: command (Argument) command: command: command: command (Argument), 52  
atom_pert_dist: command: command: command: command (Argument), 52  
atom_pert_dist (Argument), 53  
command: command: command: command: command (Argument), 52  
command: command: command: command: command (Argument), 16  
command: command: command: command: command (Argument), 15  
complete() (dp- gen2.exploration.scheduler.convergence_check_stage_sche complete() (dp- gen2.exploration.scheduler.scheduler.ExplorationScheduler method), 79  
method), 79  
complete() (dp- gen2.exploration.scheduler.scheduler.ExplorationScheduler method), 81  
method), 81  
calculate_ratio() (dp- concentration (Argument) gen2.exploration.report.naive_report.NaiveExplorationReport method), 77  
gen2.exploration.report.naive_report.NaiveExplorationReport method), 77`

### B

`BCC` (class in `dp- gen2.utils.unit_cells), 111`

### C

`calculate_ratio()`

`gen2.exploration.report.naive_report.NaiveExplorationReport`

concentration:, 53  
 concentration:  
   concentration (Argument), 53  
 ConcurrentLearning (class in dp-  
   gen2.flow.dpgen\_loop), 90  
 ConcurrentLearningBlock (class in dp-  
   gen2.superop.block), 103  
 ConcurrentLearningLoop (class in dp-  
   gen2.flow.dpgen\_loop), 90  
 ConfFilter (class in dp-  
   gen2.exploration.selector.conf\_filter), 83  
 ConfFilters (class in dp-  
   gen2.exploration.selector.conf\_filter), 83  
 config:  
   explore[lmp]/config (Argument), 16  
   fp[vasp]/config (Argument), 15  
   train[dp]/config (Argument), 18  
 config\_args() (in module dpgen2.op.run\_dp\_train), 99  
 config\_args() (in module dpgen2.op.run\_lmp), 100  
 config\_args() (in module dpgen2.op.run\_vasp), 101  
 configuration\_prefix:  
   explore[lmp]/configuration\_prefix (Argu-  
   ment), 17  
 configurations:  
   explore[lmp]/configurations (Argument), 17  
 ConfSelector (class in dp-  
   gen2.exploration.selector.conf\_selector),  
   84  
 ConfSelectorLammpsFrames (class in dp-  
   gen2.exploration.selector.conf\_selector\_frame),  
   84  
 continue\_on\_failed (Argument)  
   continue\_on\_failed:, 56  
 continue\_on\_failed:  
   continue\_on\_failed (Argument), 56  
   default\_step\_config/continue\_on\_failed  
   (Argument), 47  
   step\_configs/cl\_step\_config/continue\_on\_failed  
   (Argument), 44  
   step\_configs/collect\_data\_config/continue\_on\_failed  
   (Argument), 41  
   step\_configs/prepare\_explore\_config/continue\_on\_failed  
   (Argument), 27  
   step\_configs/prepare\_fp\_config/continue\_on\_failed  
   (Argument), 32  
   step\_configs/prepare\_train\_config/continue\_on\_failed  
   (Argument), 21  
   step\_configs/run\_explore\_config/continue\_on\_failed  
   (Argument), 29  
   step\_configs/run\_fp\_config/continue\_on\_failed  
   (Argument), 35  
   step\_configs/run\_train\_config/continue\_on\_failed  
   (Argument), 24  
   step\_configs/select\_confs\_config/continue\_on\_failed  
   (Argument), 38  
 continue\_on\_num\_success (Argument)  
   continue\_on\_num\_success:, 56  
 continue\_on\_num\_success:  
   continue\_on\_num\_success (Argument), 56  
   default\_step\_config/continue\_on\_num\_success  
   (Argument), 47  
   step\_configs/cl\_step\_config/continue\_on\_num\_success  
   (Argument), 44  
   step\_configs/collect\_data\_config/continue\_on\_num\_succe  
   (Argument), 41  
   step\_configs/prepare\_explore\_config/continue\_on\_num\_succe  
   (Argument), 27  
   step\_configs/prepare\_fp\_config/continue\_on\_num\_success  
   (Argument), 33  
   step\_configs/prepare\_train\_config/continue\_on\_num\_succe  
   (Argument), 21  
   step\_configs/run\_explore\_config/continue\_on\_num\_succe  
   (Argument), 30  
   step\_configs/run\_fp\_config/continue\_on\_num\_success  
   (Argument), 35  
   step\_configs/run\_train\_config/continue\_on\_num\_success  
   (Argument), 24  
   step\_configs/select\_confs\_config/continue\_on\_num\_succe  
   (Argument), 38  
 continue\_on\_success\_ratio (Argument)  
   continue\_on\_success\_ratio:, 56  
 continue\_on\_success\_ratio:  
   continue\_on\_success\_ratio (Argument), 56  
   default\_step\_config/continue\_on\_success\_ratio  
   (Argument), 47  
   step\_configs/cl\_step\_config/continue\_on\_success\_ratio  
   (Argument), 45  
   step\_configs/collect\_data\_config/continue\_on\_success\_r  
   (Argument), 42  
   step\_configs/prepare\_explore\_config/continue\_on\_success\_r  
   (Argument), 27  
   step\_configs/prepare\_fp\_config/continue\_on\_success\_ratio  
   (Argument), 33  
   step\_configs/prepare\_train\_config/continue\_on\_success\_ra  
   (Argument), 21  
   step\_configs/run\_explore\_config/continue\_on\_success\_ra  
   (Argument), 30  
   step\_configs/run\_fp\_config/continue\_on\_success\_ratio  
   (Argument), 36  
   step\_configs/run\_train\_config/continue\_on\_success\_rati  
   (Argument), 24  
   step\_configs/select\_confs\_config/continue\_on\_success\_r  
   (Argument), 39  
 conv\_accuracy:  
   explore[lmp]/conv\_accuracy (Argument), 16  
 converged() (dpgen2.exploration.scheduler.convergence\_check\_stage\_sch  
   method), 79  
 converged() (dpgen2.exploration.scheduler.stage\_scheduler.StageSchedul



method), 82

ConvergenceCheckStageScheduler (class in *dp-gen2.exploration.scheduler.convergence\_check\_stage\_scheduler*), 46

79

## D

decide\_init\_model() (dp-gen2.op.run\_dp\_train.RunDPTrain static method), 98

default\_step\_config (Argument) default\_step\_config:, 46

default\_step\_config/continue\_on\_failed (Argument) continue\_on\_failed:, 47

default\_step\_config/continue\_on\_num\_success (Argument) continue\_on\_num\_success:, 47

default\_step\_config/continue\_on\_success\_ratio (Argument) continue\_on\_success\_ratio:, 47

default\_step\_config/executor (Argument) executor:, 47

default\_step\_config/executor/type (Argument) type:, 48

default\_step\_config/executor[lebesgue\_v2]/extra (Argument) extra:, 48

default\_step\_config/executor[lebesgue\_v2]/extra/job\_type (Argument) job\_type:, 48

default\_step\_config/executor[lebesgue\_v2]/extra/program\_id (Argument) program\_id:, 48

default\_step\_config/executor[lebesgue\_v2]/extra/scass\_type (Argument) scass\_type:, 48

default\_step\_config/executor[lebesgue\_v2]/extra/template\_cover\_cmd\_escape\_bug (Argument) template\_cover\_cmd\_escape\_bug:, 48

default\_step\_config/parallelism (Argument) parallelism:, 47

default\_step\_config/template\_config (Argument) template\_config:, 46

default\_step\_config/template\_config/envs (Argument) envs:, 47

default\_step\_config/template\_config/image (Argument) image:, 46

default\_step\_config/template\_config/retry\_on\_transient\_error (Argument) retry\_on\_transient\_error:, 47

default\_step\_config/template\_config/timeout (Argument)

default\_step\_config/template\_config/timeout\_as\_transient\_error (Argument) timeout\_as\_transient\_error:, 47

default\_step\_config: default\_step\_config (Argument), 46

default\_step\_config\_args() (in module *dp-gen2.entrypoint.submit\_args*), 76

dflow\_conf\_args() (in module *dp-gen2.entrypoint.submit\_args*), 76

dflow\_config (Argument) dflow\_config:, 49

dflow\_config() (in module *dp-gen2.utils.dflow\_config*), 109

dflow\_config: dflow\_config (Argument), 49

dflow\_config\_lower() (in module *dp-gen2.utils.dflow\_config*), 109

dflow\_s3\_config() (in module *dp-gen2.utils.dflow\_config*), 109

dflow\_s3\_config\_lower() (in module *dp-gen2.utils.dflow\_config*), 109

DIAMOND (class in *dp-gen2.utils.unit\_cells*), 111

download() (in module *dp-gen2.entrypoint.download*), 73

download\_dp\_gen2\_artifacts() (in module *dp-gen2.utils.download\_dp\_gen2\_artifacts*), 110

DownloadDefinition (class in *dp-gen2.utils.download\_dp\_gen2\_artifacts*), 110

dp\_train\_args() (in module *dp-gen2.entrypoint.submit\_args*), 76

dp\_gen2 module, 73

dp\_gen2.constants module, 113

dp\_gen2.module\_cover\_cmd\_escape\_bug module, 73

dp\_gen2.entrypoint module, 73

dp\_gen2.entrypoint.download module, 73

dp\_gen2.entrypoint.main module, 73

dp\_gen2.entrypoint.showkey module, 74

dp\_gen2.entrypoint.status module, 74

dp\_gen2.entrypoint.submit module, 74

dp\_gen2.entrypoint.submit\_args module, 76

dp\_gen2.scheduler module, 77

dp\_gen2.entrypoint.watch module, 77

dp\_gen2.exploration

module, 77  
 dpngen2.exploration.report  
   module, 77  
 dpngen2.exploration.report.naive\_report  
   module, 77  
 dpngen2.exploration.report.report  
   module, 78  
 dpngen2.exploration.report.trajs\_report  
   module, 78  
 dpngen2.exploration.scheduler  
   module, 79  
 dpngen2.exploration.scheduler.convergence\_checker  
   module, 79  
 dpngen2.exploration.scheduler.scheduler  
   module, 80  
 dpngen2.exploration.scheduler.stage\_scheduler  
   module, 82  
 dpngen2.exploration.selector  
   module, 83  
 dpngen2.exploration.selector.conf\_filter  
   module, 83  
 dpngen2.exploration.selector.conf\_selector  
   module, 84  
 dpngen2.exploration.selector.conf\_selector\_framework  
   module, 84  
 dpngen2.exploration.selector.trust\_level  
   module, 85  
 dpngen2.exploration.task  
   module, 85  
 dpngen2.exploration.task.lmp  
   module, 85  
 dpngen2.exploration.task.lmp.lmp\_input  
   module, 85  
 dpngen2.exploration.task.npt\_task\_group  
   module, 86  
 dpngen2.exploration.task.stage  
   module, 87  
 dpngen2.exploration.task.task  
   module, 88  
 dpngen2.flow  
   module, 90  
 dpngen2.flow.dpngen\_loop  
   module, 90  
 dpngen2.fp  
   module, 93  
 dpngen2.fp.vasp  
   module, 93  
 dpngen2.op  
   module, 93  
 dpngen2.op.collect\_data  
   module, 93  
 dpngen2.op.md\_settings  
   module, 94  
 dpngen2.op.prep\_dp\_train  
   module, 95  
 dpngen2.op.prep\_lmp  
   module, 96  
 dpngen2.op.prep\_vasp  
   module, 97  
 dpngen2.op.run\_dp\_train  
   module, 98  
 dpngen2.op.run\_lmp  
   module, 99  
 dpngen2.op.run\_vasp  
   module, 100  
 dpngen2.scheduler  
   module, 102  
 dpngen2.scheduler\_confs  
   module, 102  
 dpngen2.superop  
   module, 103  
 dpngen2.superop.block  
   module, 103  
 dpngen2.superop.prep\_run\_dp\_train  
   module, 104  
 dpngen2.superop.prep\_run\_fp  
   module, 105  
 dpngen2.superop.prep\_run\_lmp  
   module, 106  
 dpngen2.utils  
   module, 107  
 dpngen2.utils.alloy\_conf  
   module, 107  
 dpngen2.utils.chdir  
   module, 108  
 dpngen2.utils.dflow\_config  
   module, 109  
 dpngen2.utils.dflow\_query  
   module, 109  
 dpngen2.utils.download\_dpngen2\_artifacts  
   module, 110  
 dpngen2.utils.obj\_artifact  
   module, 110  
 dpngen2.utils.run\_command  
   module, 110  
 dpngen2.utils.step\_config  
   module, 110  
 dpngen2.utils.unit\_cells  
   module, 111  
 dpngen\_step\_config\_args() (in module *dp-  
   gen2.entrypoint.submit\_args*), 76  
 dump\_object\_to\_file() (in module *dp-  
   gen2.utils.obj\_artifact*), 110

## E

envs:  
   default\_step\_config/template\_config/envs  
     (*Argument*), 47  
   step\_configs/cl\_step\_config/template\_config/envs  
     (*Argument*), 44

```

step_configs/collect_data_config/template_config/envs (Argument), 41
step_configs/collect_data_config/template_config/envs (Argument), 41
step_configs/prep_explore_config/template_config/envs (Argument), 26
step_configs/prep_explore_config/template_config/envs (Argument), 26
step_configs/prep_fp_config/template_config/envs (Argument), 32
step_configs/prep_fp_config/template_config/envs (Argument), 32
step_configs/prep_train_config/template_config/envs (Argument), 21
step_configs/prep_train_config/template_config/envs (Argument), 21
step_configs/run_explore_config/template_config/envs (Argument), 29
step_configs/run_explore_config/template_config/envs (Argument), 29
step_configs/run_fp_config/template_config/envs (Argument), 35
step_configs/run_fp_config/template_config/envs (Argument), 35
step_configs/run_train_config/template_config/envs (Argument), 23
step_configs/run_train_config/template_config/envs (Argument), 23
step_configs/select_confs_config/template_config/envs (Argument), 38
step_configs/select_confs_config/template_config/envs (Argument), 38
template_config/envs (Argument), 57
template_config/envs (Argument), 57
execute() (dpngen2.flow.dpgen_loop.MakeBlockId method), 92
execute() (dpngen2.flow.dpgen_loop.SchedulerWrapper method), 92
execute() (dpngen2.op.collect_data.CollectData method), 94
execute() (dpngen2.op.prep_dp_train.PrepDPTrain method), 95
execute() (dpngen2.op.prep_lmp.PrepLmp method), 96
execute() (dpngen2.op.prep_vasp.PrepVasp method), 97
execute() (dpngen2.op.run_dp_train.RunDPTrain method), 98
execute() (dpngen2.op.run_lmp.RunLmp method), 99
execute() (dpngen2.op.run_vasp.RunVasp method), 101
execute() (dpngen2.op.select_confs.SelectConfs method), 102
executor (Argument)
  executor:, 55
executor/type (Argument)
  type:, 55
executor:
  default_step_config/executor (Argument), 47
  executor (Argument), 55
  step_configs/cl_step_config/executor (Argument), 45
  step_configs/collect_data_config/executor (Argument), 42
  step_configs/prep_explore_config/executor (Argument), 27
  step_configs/prep_fp_config/executor (Argument), 33
  step_configs/prep_train_config/executor (Argument), 21
  step_configs/run_explore_config/executor (Argument), 30
  step_configs/run_fp_config/executor (Argument), 36
  step_configs/run_train_config/executor (Argument), 24
  step_configs/select_confs_config/executor (Argument), 39
  executor[lebesgue_v2]/extra (Argument)
    extra:, 55
  executor[lebesgue_v2]/extra/job_type (Argument)
    job_type:, 55
  executor[lebesgue_v2]/extra/program_id (Argument)
    program_id:, 55
  executor[lebesgue_v2]/extra/scass_type (Argument)
    scass_type:, 55
  executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug (Argument)
    template_cover_cmd_escape_bug:, 56
  expand_idx() (in module dpngen2.entrypoint.submit), 74
  expand_sys_str() (in module dpngen2.entrypoint.submit), 74
  ExplorationReport (class in dpngen2.exploration.report.report), 78
  ExplorationScheduler (class in dpngen2.exploration.scheduler.scheduler), 80
  ExplorationStage (class in dpngen2.exploration.task.stage), 87
  ExplorationTask (class in dpngen2.exploration.task.task), 88
  ExplorationTaskGroup (class in dpngen2.exploration.task.task), 88
  explore (Argument)
    explore:, 16
  explore/type (Argument)
    type:, 16
  explore:
    explore (Argument), 16
  explore[lmp]/config (Argument)
    config:, 16
  explore[lmp]/config/command (Argument)
    command:, 16
  explore[lmp]/configuration_prefix (Argument)
    configuration_prefix:, 17
  explore[lmp]/configurations (Argument)
    configurations:, 17
  explore[lmp]/conv_accuracy (Argument)
    conv_accuracy:, 16
  explore[lmp]/f_trust_hi (Argument)
    f_trust_hi:, 17
  explore[lmp]/f_trust_lo (Argument)
    f_trust_lo:, 17
  explore[lmp]/fatal_at_max (Argument)
    fatal_at_max:, 17
  explore[lmp]/max_numb_iter (Argument)

```

```

    max_numb_iter:, 16
explore[lmp]/stages (Argument)
    stages:, 17
explore[lmp]/v_trust_hi (Argument)
    v_trust_hi:, 17
explore[lmp]/v_trust_lo (Argument)
    v_trust_lo:, 17
extra:
    default_step_config/executor[lebesgue_v2]/fp[vasp]/config (Argument)
        (Argument), 48
    executor[lebesgue_v2]/extra (Argument), 55
    step_configs/cl_step_config/executor[lebesgue_v2]/extra
        (Argument), 45
    step_configs/collect_data_config/executor[lebesgue_v2]/extra
        (Argument), 42
    step_configs/prepare_explore_config/executor[lebesgue_v2]/extra
        (Argument), 27
    step_configs/prepare_fp_config/executor[lebesgue_v2]/extra
        (Argument), 33
    step_configs/prepare_train_config/executor[lebesgue_v2]/extra
        (Argument), 22
    step_configs/run_explore_config/executor[lebesgue_v2]/extra
        (Argument), 30
    step_configs/run_fp_config/executor[lebesgue_v2]/extra
        (Argument), 36
    step_configs/run_train_config/executor[lebesgue_v2]/extra
        (Argument), 24
    step_configs/select_confs_config/executor[lebesgue_v2]/extra
        (Argument), 39
F
f_trust_hi:
    explore[lmp]/f_trust_hi (Argument), 17
f_trust_lo:
    explore[lmp]/f_trust_lo (Argument), 17
failed_ratio() (dpngen2.exploration.report.naive_report.NaiveExplorationReport
    method), 77
failed_ratio() (dpngen2.exploration.report.report.ExplorationReport
    method), 78
failed_ratio() (dpngen2.exploration.report.trajs_report.TrajsExplorationReport
    method), 78
fatal_at_max:
    explore[lmp]/fatal_at_max (Argument), 17
FCC (class in dpngen2.utils.unit_cells), 111
files() (dpngen2.exploration.task.task.ExplorationTask
    method), 88
find_slice_ranges() (in module dp-
    gen2.utils.dflow_query), 109
fmt (Argument)
    fmt:, 53
fmt:
    fmt (Argument), 53
FooTask (class in dpngen2.exploration.task.task), 89
FooTaskGroup (class in dpngen2.exploration.task.task),
    89
fp (Argument)
    fp:, 15
fp/type (Argument)
    type:, 15
fp:
    fp (Argument), 15
fp[vasp]/config (Argument)
    config:, 15
fp[vasp]/config/command (Argument)
    command:, 15
fp[vasp]/config/log (Argument)
    log:, 15
fp[vasp]/config/out (Argument)
    out:, 15
fp[vasp]/incar (Argument)
    incar:, 15
fp[vasp]/pp_files (Argument)
    pp_files:, 15
fp[vasp]/task_max (Argument)
    task_max:, 15
G
gen_box() (dpngen2.utils.unit_cells.BCC method), 111
gen_box() (dpngen2.utils.unit_cells.DIAMOND method),
    111
gen_box() (dpngen2.utils.unit_cells.FCC method), 111
gen_box() (dpngen2.utils.unit_cells.HCP method), 112
gen_box() (dpngen2.utils.unit_cells.SC method), 112
gen_doc() (in module dpngen2.entrypoint.submit_args),
    76
gen_doc() (in module dpngen2.utils.alloy_conf), 108
gen_doc() (in module dpngen2.utils.step_config), 110
generate_alloy_conf_args() (in module dp-
    gen2.utils.alloy_conf), 108
generate_alloy_conf_file_content() (in module
    dpngen2.utils.alloy_conf), 108
generate_file_content() (dp-
    gen2.utils.alloy_conf.AlloyConf method),
    107
generate_systems() (dp-
    gen2.utils.alloy_conf.AlloyConf method),
    107
generate_unit_cell() (in module dp-
    gen2.utils.unit_cells), 112
get_candidates() (dp-
    gen2.exploration.report.trajs_report.TrajsExplorationReport
    method), 78
get_convergence_ratio() (dp-
    gen2.exploration.scheduler.scheduler.ExplorationScheduler
    method), 81
get_input_sign() (dp-
    gen2.flow.dpgen_loop.MakeBlockId class

```

method), 92

get\_input\_sign() (dp-gen2.flow.dpgen\_loop.SchedulerWrapper class method), 92

get\_input\_sign() (dp-gen2.op.collect\_data.CollectData class method), 94

get\_input\_sign() (dp-gen2.op.prep\_dp\_train.PrepDPTrain class method), 96

get\_input\_sign() (dp-gen2.op.prep\_lmp.PrepLmp class method), 96

get\_input\_sign() (dp-gen2.op.prep\_vasp.PrepVasp class method), 97

get\_input\_sign() (dp-gen2.op.run\_dp\_train.RunDPTrain class method), 99

get\_input\_sign() (dp-gen2.op.run\_lmp.RunLmp class method), 100

get\_input\_sign() (dp-gen2.op.run\_vasp.RunVasp class method), 101

get\_input\_sign() (dp-gen2.op.select\_confs.SelectConfs class method), 102

get\_iteration() (dp-gen2.exploration.scheduler.scheduler.ExplorationScheduler method), 81

get\_iteration() (in module dp-gen2.utils.dflow\_query), 109

get\_kspacing\_kgamma\_from\_incar() (in module dp-gen2.entrypoint.submit), 74

get\_last\_iteration() (in module dp-gen2.utils.dflow\_query), 109

get\_last\_scheduler() (in module dp-gen2.utils.dflow\_query), 109

get\_output\_sign() (dp-gen2.flow.dpgen\_loop.MakeBlockId class method), 92

get\_output\_sign() (dp-gen2.flow.dpgen\_loop.SchedulerWrapper class method), 92

get\_output\_sign() (dp-gen2.op.collect\_data.CollectData class method), 94

get\_output\_sign() (dp-gen2.op.prep\_dp\_train.PrepDPTrain class method), 96

get\_output\_sign() (dp-gen2.op.prep\_lmp.PrepLmp class method), 97

get\_output\_sign() (dp-gen2.op.prep\_vasp.PrepVasp class method), 98

get\_output\_sign() (dp-gen2.op.run\_dp\_train.RunDPTrain class method), 99

get\_output\_sign() (dp-gen2.op.run\_lmp.RunLmp class method), 100

get\_output\_sign() (dp-gen2.op.run\_vasp.RunVasp class method), 101

get\_output\_sign() (dp-gen2.op.select\_confs.SelectConfs class method), 102

get\_stage() (dp-gen2.exploration.scheduler.scheduler.ExplorationScheduler method), 81

get\_stage\_of\_iterations() (dp-gen2.exploration.scheduler.scheduler.ExplorationScheduler method), 81

get\_subkey() (in module dp-gen2.utils.dflow\_query), 109

## H

HCP (class in dp-gen2.utils.unit\_cells), 112

## I

image:

- default\_step\_config/template\_config/image (Argument), 46
- step\_configs/cl\_step\_config/template\_config/image (Argument), 43
- step\_configs/collect\_data\_config/template\_config/image (Argument), 40
- step\_configs/prep\_explore\_config/template\_config/image (Argument), 26
- step\_configs/prep\_fp\_config/template\_config/image (Argument), 32
- step\_configs/prep\_train\_config/template\_config/image (Argument), 20
- step\_configs/run\_explore\_config/template\_config/image (Argument), 29
- step\_configs/run\_fp\_config/template\_config/image (Argument), 35
- step\_configs/run\_train\_config/template\_config/image (Argument), 23
- step\_configs/select\_confs\_config/template\_config/image (Argument), 37
- template\_config/image (Argument), 56

incard:

- fp[vasp]/incard (Argument), 16

incard\_from\_file() (dp-gen2.fp.vasp.VaspInputs method), 93

incard\_template (dp-gen2.fp.vasp.VaspInputs property), 93

init\_data\_prefix:

- inputs/init\_data\_prefix (Argument), 19

init\_data\_sys:

- inputs/init\_data\_sys (Argument), 19

init\_executor() (in module dp-gen2.utils.step\_config), 110



`init_keys(dp-gen2.flow.dp-gen-loop.ConcurrentLearning property), 90`  
`init_model_numb_steps (Argument)`  
`init_model_numb_steps:, 51`  
`init_model_numb_steps:`  
`init_model_numb_steps (Argument), 51`  
`train[dp]/config/init_model_numb_steps (Argument), 18`  
`init_model_old_ratio (Argument)`  
`init_model_old_ratio:, 51`  
`init_model_old_ratio:`  
`init_model_old_ratio (Argument), 51`  
`train[dp]/config/init_model_old_ratio (Argument), 18`  
`init_model_policy (Argument)`  
`init_model_policy:, 51`  
`init_model_policy:`  
`init_model_policy (Argument), 51`  
`train[dp]/config/init_model_policy (Argument), 18`  
`init_model_start_lr (Argument)`  
`init_model_start_lr:, 51`  
`init_model_start_lr:`  
`init_model_start_lr (Argument), 51`  
`train[dp]/config/init_model_start_lr (Argument), 18`  
`init_model_start_pref_e (Argument)`  
`init_model_start_pref_e:, 51`  
`init_model_start_pref_e:`  
`init_model_start_pref_e (Argument), 51`  
`train[dp]/config/init_model_start_pref_e (Argument), 18`  
`init_model_start_pref_f (Argument)`  
`init_model_start_pref_f:, 51`  
`init_model_start_pref_f:`  
`init_model_start_pref_f (Argument), 51`  
`train[dp]/config/init_model_start_pref_f (Argument), 19`  
`init_model_start_pref_v (Argument)`  
`init_model_start_pref_v:, 51`  
`init_model_start_pref_v:`  
`init_model_start_pref_v (Argument), 51`  
`train[dp]/config/init_model_start_pref_v (Argument), 19`  
`input_args() (in module dp-gen2.entrypoint.submit_args), 76`  
`input_artifacts (dp-gen2.flow.dp-gen-loop.ConcurrentLearning property), 90`  
`input_artifacts (dp-gen2.flow.dp-gen-loop.ConcurrentLearningLoop property), 91`  
`input_artifacts (dp-gen2.superop.block.ConcurrentLearningBlock property), 103`  
`input_artifacts (dp-gen2.superop.prep-run-dp-train.PrepRunDPTrain property), 104`  
`input_artifacts (dp-gen2.superop.prep-run-fp.PrepRunFp property), 105`  
`input_artifacts (dp-gen2.superop.prep-run-lmp.PrepRunLmp property), 106`  
`input_parameters (dp-gen2.flow.dp-gen-loop.ConcurrentLearning property), 90`  
`input_parameters (dp-gen2.flow.dp-gen-loop.ConcurrentLearningLoop property), 91`  
`input_parameters (dp-gen2.superop.block.ConcurrentLearningBlock property), 103`  
`input_parameters (dp-gen2.superop.prep-run-dp-train.PrepRunDPTrain property), 104`  
`input_parameters (dp-gen2.superop.prep-run-fp.PrepRunFp property), 105`  
`input_parameters (dp-gen2.superop.prep-run-lmp.PrepRunLmp property), 106`  
`inputs (Argument)`  
`inputs:, 19`  
`inputs/init_data_prefix (Argument)`  
`init_data_prefix:, 19`  
`inputs/init_data_sys (Argument)`  
`init_data_sys:, 19`  
`inputs/mass_map (Argument)`  
`mass_map:, 19`  
`inputs/type_map (Argument)`  
`type_map:, 19`  
`inputs:`  
`inputs (Argument), 19`

## J

`job_type:`  
`default_step_config/executor[lebesgue_v2]/extra/job_type (Argument), 48`  
`executor[lebesgue_v2]/extra/job_type (Argument), 55`  
`step_configs/cl_step_config/executor[lebesgue_v2]/extra (Argument), 46`  
`step_configs/collect_data_config/executor[lebesgue_v2] (Argument), 43`  
`step_configs/prepare_explore_config/executor[lebesgue_v2] (Argument), 28`

step\_configs/prep\_fp\_config/executor[lebesgue\_v2]/extra/job\_type  
 (Argument), 34 log:, 52  
 step\_configs/prep\_train\_config/executor[lebesgue\_v2]/extra/job\_type  
 (Argument), 22 fp[vasp]/config/log (Argument), 15  
 step\_configs/run\_explore\_config/executor[lebesgue\_v2]/extra/job\_type  
 (Argument), 31 loop\_keys (dpngen2.flow.dpngen\_loop.ConcurrentLearning  
 step\_configs/run\_fp\_config/executor[lebesgue\_v2]/extra/job\_type  
 (Argument), 37  
 step\_configs/run\_train\_config/executor[lebesgue\_v2]/extra/job\_type  
 (Argument), 25  
 step\_configs/select\_confs\_config/executor[lebesgue\_v2]/extra/job\_type  
 (Argument), 40

## K

keys (dpngen2.flow.dpngen\_loop.ConcurrentLearningLoop  
 property), 91  
 keys (dpngen2.superop.block.ConcurrentLearningBlock  
 property), 103  
 keys (dpngen2.superop.prep\_run\_dp\_train.PrepRunDPTrain  
 property), 104  
 keys (dpngen2.superop.prep\_run\_fp.PrepRunFp prop-  
 erty), 105  
 keys (dpngen2.superop.prep\_run\_lmp.PrepRunLmp prop-  
 erty), 106

## L

lattice (Argument)  
 lattice:, 53  
 lattice:  
 lattice (Argument), 53  
 lebesgue\_conf\_args() (in module dp-  
 gen2.entrypoint.submit\_args), 76  
 lebesgue\_context\_config (Argument)  
 lebesgue\_context\_config:, 49  
 lebesgue\_context\_config:  
 lebesgue\_context\_config (Argument), 49  
 lebesgue\_executor\_args() (in module dp-  
 gen2.utils.step\_config), 110  
 lebesgue\_extra\_args() (in module dp-  
 gen2.utils.step\_config), 110  
 level\_f\_hi (dpngen2.exploration.selector.trust\_level.TrustLevel  
 property), 85  
 level\_f\_lo (dpngen2.exploration.selector.trust\_level.TrustLevel  
 property), 85  
 level\_v\_hi (dpngen2.exploration.selector.trust\_level.TrustLevel  
 property), 85  
 level\_v\_lo (dpngen2.exploration.selector.trust\_level.TrustLevel  
 property), 85  
 lmp\_args() (dpngen2.op.run\_lmp.RunLmp static  
 method), 100  
 lmp\_args() (in module dpngen2.entrypoint.submit\_args),  
 76  
 load\_object\_from\_file() (in module dp-  
 gen2.utils.obj\_artifact), 110

main() (in module dpngen2.entrypoint.main), 73  
 main\_parser() (in module dpngen2.entrypoint.main), 73  
 make\_concurrent\_learning\_op() (in module dp-  
 gen2.entrypoint.submit), 74  
 make\_conf\_list() (in module dp-  
 gen2.entrypoint.submit), 76  
 make\_kpoints() (dpngen2.fp.vasp.VaspInputs method),  
 93  
 make\_kspacing\_kpoints() (in module dp-  
 gen2.fp.vasp), 93  
 make\_lmp\_input() (in module dp-  
 gen2.exploration.task.lmp.lmp\_input), 85  
 make\_naive\_exploration\_scheduler() (in module  
 dpngen2.entrypoint.submit), 76  
 make\_potcar() (dpngen2.fp.vasp.VaspInputs method), 93  
 make\_task() (dpngen2.exploration.task.npt\_task\_group.NPTTaskGroup  
 method), 86  
 make\_task() (dpngen2.exploration.task.stage.ExplorationStage  
 method), 87  
 MakeBlockId (class in dpngen2.flow.dpngen\_loop), 91  
 mass\_map:  
 inputs/mass\_map (Argument), 19  
 matched\_step\_key() (in module dp-  
 gen2.utils.dflow\_query), 109  
 max\_numb\_iter:  
 explore[lmp]/max\_numb\_iter (Argument), 16  
 MDSettings (class in dpngen2.op.md\_settings), 94  
 module  
 dpngen2, 73  
 dpngen2.constants, 112  
 dpngen2.entrypoint, 73  
 dpngen2.entrypoint.download, 73  
 dpngen2.entrypoint.main, 73  
 dpngen2.entrypoint.showkey, 74  
 dpngen2.entrypoint.status, 74  
 dpngen2.entrypoint.submit, 74  
 dpngen2.entrypoint.submit\_args, 76  
 dpngen2.entrypoint.watch, 77  
 dpngen2.exploration, 77  
 dpngen2.exploration.report, 77  
 dpngen2.exploration.report.naive\_report,  
 77  
 dpngen2.exploration.report.report, 78  
 dpngen2.exploration.report.trajs\_report,  
 78

dpngen2.exploration.scheduler, 79  
 dpngen2.exploration.scheduler.convergence\_check\_stage\_scheduler, 79  
 dpngen2.exploration.scheduler.scheduler, 80  
 dpngen2.exploration.scheduler.stage\_scheduler, 82  
 dpngen2.exploration.selector, 83  
 dpngen2.exploration.selector.conf\_filter, 83  
 dpngen2.exploration.selector.conf\_selector, 84  
 dpngen2.exploration.selector.conf\_selector\_frame, 84  
 dpngen2.exploration.selector.trust\_level, 85  
 dpngen2.exploration.task, 85  
 dpngen2.exploration.task.lmp, 85  
 dpngen2.exploration.task.lmp.lmp\_input, 85  
 dpngen2.exploration.task.npt\_task\_group, 86  
 dpngen2.exploration.task.stage, 87  
 dpngen2.exploration.task.task, 88  
 dpngen2.flow, 90  
 dpngen2.flow.dpngen\_loop, 90  
 dpngen2.fp, 93  
 dpngen2.fp.vasp, 93  
 dpngen2.op, 93  
 dpngen2.op.collect\_data, 93  
 dpngen2.op.md\_settings, 94  
 dpngen2.op.prep\_dp\_train, 95  
 dpngen2.op.prep\_lmp, 96  
 dpngen2.op.prep\_vasp, 97  
 dpngen2.op.run\_dp\_train, 98  
 dpngen2.op.run\_lmp, 99  
 dpngen2.op.run\_vasp, 100  
 dpngen2.op.select\_confs, 102  
 dpngen2.superop, 103  
 dpngen2.superop.block, 103  
 dpngen2.superop.prep\_run\_dp\_train, 104  
 dpngen2.superop.prep\_run\_fp, 105  
 dpngen2.superop.prep\_run\_lmp, 106  
 dpngen2.utils, 107  
 dpngen2.utils.alloy\_conf, 107  
 dpngen2.utils.chdir, 108  
 dpngen2.utils.dflow\_config, 109  
 dpngen2.utils.dflow\_query, 109  
 dpngen2.utils.download\_dpngen2\_artifacts, 110  
 dpngen2.utils.obj\_artifact, 110  
 dpngen2.utils.run\_command, 110  
 dpngen2.utils.step\_config, 110  
 dpngen2.utils.unit\_cells, 111

## N

NaiveExplorationReport (class in dp-  
 gen2.exploration.report.naive\_report), 77  
 normalize() (in module dp-  
 gen2.entrypoint.submit\_args), 76  
 normalize() (in module dpngen2.utils.alloy\_conf), 108  
 normalize() (in module dpngen2.utils.step\_config), 110  
 normalize\_config() (dp-  
 gen2.op.run\_dp\_train.RunDPTrain static  
 method), 99  
 normalize\_config() (dpngen2.op.run\_lmp.RunLmp  
 static method), 100  
 normalize\_config() (dpngen2.op.run\_vasp.RunVasp  
 static method), 101  
 NPTTaskGroup (class in dp-  
 gen2.exploration.task.npt\_task\_group), 86  
 numb\_atoms() (dpngen2.utils.unit\_cells.BCC method),  
 111  
 numb\_atoms() (dpngen2.utils.unit\_cells.DIAMOND  
 method), 111  
 numb\_atoms() (dpngen2.utils.unit\_cells.FCC method),  
 111  
 numb\_atoms() (dpngen2.utils.unit\_cells.HCP method),  
 112  
 numb\_atoms() (dpngen2.utils.unit\_cells.SC method), 112  
 numb\_confs (Argument)  
 numb\_confs:, 53  
 numb\_confs:  
 numb\_confs (Argument), 53  
 numb\_models:  
 train[dp]/numb\_models (Argument), 19

## O

out (Argument)  
 out:, 52  
 out:  
 fp[vasp]/config/out (Argument), 15  
 out (Argument), 52  
 output\_artifacts (dp-  
 gen2.flow.dpngen\_loop.ConcurrentLearning  
 property), 90  
 output\_artifacts (dp-  
 gen2.flow.dpngen\_loop.ConcurrentLearningLoop  
 property), 91  
 output\_artifacts (dp-  
 gen2.superop.block.ConcurrentLearningBlock  
 property), 103  
 output\_artifacts (dp-  
 gen2.superop.prep\_run\_dp\_train.PrepRunDPTrain  
 property), 104  
 output\_artifacts (dp-  
 gen2.superop.prep\_run\_fp.PrepRunFp prop-  
 erty), 105



output\_artifacts (dp- gen2.superop.prep\_run\_imp.PrepRunLmp property), 106  
 output\_parameters (dp- gen2.flow.dpgen\_loop.ConcurrentLearning property), 90  
 output\_parameters (dp- gen2.flow.dpgen\_loop.ConcurrentLearningLoop property), 91  
 output\_parameters (dp- gen2.superop.block.ConcurrentLearningBlock property), 103  
 output\_parameters (dp- gen2.superop.prep\_run\_dp\_train.PrepRunDPTrain property), 104  
 output\_parameters (dp- gen2.superop.prep\_run\_fp.PrepRunFp property), 105  
 output\_parameters (dp- gen2.superop.prep\_run\_imp.PrepRunLmp property), 106  
**P**  
 parallelism (Argument)  
   parallelism:, 56  
 parallelism:  
   default\_step\_config/parallelism (Argument), 47  
   parallelism (Argument), 56  
   step\_configs/cl\_step\_config/parallelism (Argument), 45  
   step\_configs/collect\_data\_config/parallelism (Argument), 42  
   step\_configs/prepare\_explore\_config/parallelism (Argument), 27  
   step\_configs/prepare\_fp\_config/parallelism (Argument), 33  
   step\_configs/prepare\_train\_config/parallelism (Argument), 21  
   step\_configs/run\_explore\_config/parallelism (Argument), 30  
   step\_configs/run\_fp\_config/parallelism (Argument), 36  
   step\_configs/run\_train\_config/parallelism (Argument), 24  
   step\_configs/select\_confs\_config/parallelism (Argument), 39  
 parse\_args() (in module dpngen2.entrypoint.main), 73  
 plan\_next\_iteration() (dp- gen2.exploration.scheduler.convergence\_check\_stage\_scheduler method), 80  
 plan\_next\_iteration() (dp- gen2.exploration.scheduler.scheduler.ExplorationScheduler method), 81  
 plan\_next\_iteration() (dp- gen2.exploration.scheduler.stage\_scheduler.StageScheduler method), 82  
 poscar\_unit() (dpngen2.utils.unit\_cells.BCC method), 111  
 poscar\_unit() (dpngen2.utils.unit\_cells.DIAMOND method), 111  
 poscar\_unit() (dpngen2.utils.unit\_cells.FCC method), 111  
 poscar\_unit() (dpngen2.utils.unit\_cells.HCP method), 112  
 poscar\_unit() (dpngen2.utils.unit\_cells.SC method), 112  
 potcars (dpngen2.fp.vasp.VaspInputs property), 93  
 potcars\_from\_file() (dpngen2.fp.vasp.VaspInputs method), 93  
 pp\_files:  
   fp[vasp]/pp\_files (Argument), 16  
 prep\_explore\_config:  
   step\_configs/prepare\_explore\_config (Argument), 25  
 prep\_fp\_config:  
   step\_configs/prepare\_fp\_config (Argument), 31  
 prep\_train\_config:  
   step\_configs/prepare\_train\_config (Argument), 20  
 PrepDPTrain (class in dpngen2.op.prep\_dp\_train), 95  
 PrepExplorationTaskGroup (in module dp- gen2.op.prep\_imp), 96  
 PrepLmp (class in dpngen2.op.prep\_imp), 96  
 PrepRunDPTrain (class in dp- gen2.superop.prep\_run\_dp\_train), 104  
 PrepRunFp (class in dpngen2.superop.prep\_run\_fp), 105  
 PrepRunLmp (class in dpngen2.superop.prep\_run\_imp), 106  
 PrepVasp (class in dpngen2.op.prep\_vasp), 97  
 print\_convergence() (dp- gen2.exploration.scheduler.scheduler.ExplorationScheduler method), 81  
 print\_keys\_in\_nice\_format() (in module dp- gen2.utils.dflow\_query), 109  
 print\_list\_steps() (in module dp- gen2.entrypoint.submit), 76  
 program\_id:  
   default\_step\_config/executor[lebesgue\_v2]/extra/program\_id (Argument), 48  
   executor[lebesgue\_v2]/extra/program\_id (Argument), 55  
   step\_configs/cl\_step\_config/executor[lebesgue\_v2]/extra/program\_id (Argument), 46  
   step\_configs/collect\_data\_config/executor[lebesgue\_v2]/extra/program\_id (Argument), 43  
   step\_configs/prepare\_explore\_config/executor[lebesgue\_v2]/extra/program\_id (Argument), 28

step\_configs/prep\_fp\_config/executor[lebesgue\_v2]/extra/program\_id  
 (Argument), 34  
 step\_configs/prep\_train\_config/executor[lebesgue\_v2]/extra/program\_id  
 (Argument), 22  
 step\_configs/run\_explore\_config/executor[lebesgue\_v2]/extra/program\_id  
 (Argument), 31  
 step\_configs/run\_fp\_config/executor[lebesgue\_v2]/extra/program\_id  
 (Argument), 37  
 step\_configs/run\_train\_config/executor[lebesgue\_v2]/extra/program\_id  
 (Argument), 25  
 step\_configs/select\_confs\_config/executor[lebesgue\_v2]/extra/program\_id  
 (Argument), 40  
 RunB3Main (class in dpngen2.op.run\_b3main), 98  
 RunLmp (class in dpngen2.op.run\_lmp), 99  
 RunVasp (class in dpngen2.op.run\_vasp), 100

## R

ratio() (dpngen2.exploration.report.naive\_report.NaiveExplorationReport  
 method), 77  
 reached\_max\_iteration() (dp-  
 gen2.exploration.scheduler.convergence\_check\_stage\_scheduler.ConvergenceCheckStageScheduler  
 method), 80  
 record\_one\_traj() (dp-  
 gen2.exploration.selector.conf\_selector\_frame.ConfSelectorFrame  
 method), 84  
 record\_traj() (dpngen2.exploration.report.trajs\_report.TrajsExplorationReport  
 method), 78  
 replicate (Argument)  
 replicate:, 53  
 replicate (Argument), 53  
 resubmit\_concurrent\_learning() (in module dp-  
 gen2.entrpoint.submit), 76  
 retry\_on\_transient\_error:  
 default\_step\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 47  
 step\_configs/cl\_step\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 44  
 step\_configs/collect\_data\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 41  
 step\_configs/prepare\_explore\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 26  
 step\_configs/prepare\_fp\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 32  
 step\_configs/prepare\_train\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 20  
 step\_configs/run\_explore\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 29  
 step\_configs/run\_fp\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 35  
 step\_configs/run\_train\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 23  
 step\_configs/select\_confs\_config/template\_config/retry\_on\_transient\_error  
 (Argument), 38  
 template\_config/retry\_on\_transient\_error  
 (Argument), 56

## S

s3\_config (Argument)  
 s3\_config:, 49  
 s3\_config (Argument), 49  
 SC (class in dpngen2.utils.unit\_cells), 112  
 ScissorsPumpsFrames  
 default\_step\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 48  
 executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 55  
 step\_configs/cl\_step\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 45  
 step\_configs/collect\_data\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 42  
 step\_configs/prepare\_explore\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 28  
 step\_configs/prepare\_fp\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 34  
 step\_configs/prepare\_train\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 22  
 step\_configs/run\_explore\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 31  
 step\_configs/run\_fp\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 36  
 step\_configs/run\_train\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 25  
 step\_configs/select\_confs\_config/executor[lebesgue\_v2]/extra/scass\_type  
 (Argument), 39  
 SingleRetryWrapper (class in dpngen2.flow.dpgen\_loop), 92  
 selector (in dpngen2.exploration.selector.conf\_selector.ConfSelector  
 method), 84  
 selector (in dpngen2.exploration.selector.conf\_selector\_frame.ConfSelectorFrame  
 method), 84  
 select\_confs\_config/retry\_on\_transient\_error  
 (Argument), 37  
 SelectConfs (class in dpngen2.op.select\_confs), 102

`set_conf()` (*dpngen2.exploration.task.npt\_task\_group.NPTTaskGroup* (Argument)  
 method), 86  
`set_directory()` (in module *dpngen2.utils.chdir*), 108  
`set_md()` (*dpngen2.exploration.task.npt\_task\_group.NPTTaskGroup* (Argument)  
 method), 87  
`showkey()` (in module *dpngen2.entrypoint.showkey*), 74  
`sort_slice_ops()` (in module *dp-  
 gen2.utils.dflow\_query*), 109  
`stages:`  
   `explore[lmp]/stages` (Argument), 17  
`StageScheduler` (class in *dp-  
 gen2.exploration.scheduler.stage\_scheduler*),  
 82  
`status()` (in module *dpngen2.entrypoint.status*), 74  
`step_conf_args()` (in module *dp-  
 gen2.utils.step\_config*), 110  
`step_configs` (Argument)  
   `step_configs:`, 20  
`step_configs/cl_step_config` (Argument)  
   `cl_step_config:`, 43  
`step_configs/cl_step_config/continue_on_failed`  
   (Argument)  
     `continue_on_failed:`, 44  
`step_configs/cl_step_config/continue_on_num_success` (Argument)  
   (Argument)  
     `continue_on_num_success:`, 44  
`step_configs/cl_step_config/continue_on_success_ratio` (Argument)  
   (Argument)  
     `continue_on_success_ratio:`, 45  
`step_configs/cl_step_config/executor` (Argu-  
 ment)  
   `executor:`, 45  
`step_configs/cl_step_config/executor/type`  
   (Argument)  
     `type:`, 45  
`step_configs/cl_step_config/executor[lebesgue_v2]/extra` (Argument)  
   (Argument)  
     `extra:`, 45  
`step_configs/cl_step_config/executor[lebesgue_v2]/extra/job_type`  
   (Argument)  
     `job_type:`, 46  
`step_configs/cl_step_config/executor[lebesgue_v2]/extra/program_id`  
   (Argument)  
     `program_id:`, 46  
`step_configs/cl_step_config/executor[lebesgue_v2]/extra/scass_type`  
   (Argument)  
     `scass_type:`, 45  
`step_configs/cl_step_config/executor[lebesgue_v2]/extra/template`  
   (Argument)  
     `template:`, 45  
   `template_cover_cmd_escape_bug:`, 46  
`step_configs/cl_step_config/parallelism`  
   (Argument)  
     `parallelism:`, 45  
`step_configs/cl_step_config/template_config`  
   (Argument)  
     `template_config:`, 43  
`step_configs/cl_step_config/template_config/envs`  
   (Argument)  
     `envs:`, 44  
`step_configs/cl_step_config/template_config/image`  
   (Argument)  
     `image:`, 43  
`step_configs/cl_step_config/template_config/retry_on_trans`  
   (Argument)  
     `retry_on_transient_error:`, 44  
`step_configs/cl_step_config/template_config/timeout`  
   (Argument)  
     `timeout:`, 44  
`step_configs/cl_step_config/template_config/timeout_as_tra`  
   (Argument)  
     `timeout_as_transient_error:`, 44  
`step_configs/collect_data_config` (Argument)  
   `collect_data_config:`, 40  
`step_configs/collect_data_config/continue_on_failed`  
   (Argument)  
     `continue_on_failed:`, 41  
`step_configs/collect_data_config/continue_on_num_success`  
   (Argument)  
     `continue_on_num_success:`, 41  
`step_configs/collect_data_config/continue_on_success_ratio`  
   (Argument)  
     `continue_on_success_ratio:`, 42  
`step_configs/collect_data_config/executor`  
   (Argument)  
     `executor:`, 42  
`step_configs/collect_data_config/executor/type`  
   (Argument)  
     `type:`, 42  
`step_configs/collect_data_config/executor[lebesgue_v2]/extra`  
   (Argument)  
     `extra:`, 42  
`step_configs/collect_data_config/executor[lebesgue_v2]/extra/job_type`  
   (Argument)  
     `job_type:`, 43  
`step_configs/collect_data_config/executor[lebesgue_v2]/extra/program_id`  
   (Argument)  
     `program_id:`, 43  
`step_configs/collect_data_config/executor[lebesgue_v2]/extra/scass_type`  
   (Argument)  
     `scass_type:`, 42  
`step_configs/collect_data_config/executor[lebesgue_v2]/extra/template`  
   (Argument)  
     `template:`, 45  
   `template_cover_cmd_escape_bug:`, 43  
`step_configs/collect_data_config/parallelism`  
   (Argument)  
     `parallelism:`, 42  
`step_configs/collect_data_config/template_config`  
   (Argument)

template_config:, 40	step_configs/prepare_explore_config/template_config/envs
step_configs/collect_data_config/template_config/envs (Argument)	envs:, 26
(Argument)	step_configs/prepare_explore_config/template_config/image
envs:, 41	(Argument)
step_configs/collect_data_config/template_config/image (Argument)	image:, 26
(Argument)	step_configs/prepare_explore_config/template_config/retry_on_transient_error
image:, 40	retry_on_transient_error:, 26
step_configs/collect_data_config/template_config/retry_on_transient_error (Argument)	step_configs/prepare_explore_config/template_config/timeout
(Argument)	timeout:, 26
retry_on_transient_error:, 41	step_configs/prepare_explore_config/template_config/timeout_as_transient_error
step_configs/collect_data_config/template_config/timeout (Argument)	timeout_as_transient_error:, 26
(Argument)	step_configs/prepare_fp_config (Argument)
timeout:, 41	prep_fp_config:, 31
step_configs/collect_data_config/template_config/timeout_as_transient_error (Argument)	step_configs/prepare_fp_config/continue_on_failed
(Argument)	continue_on_failed:, 32
timeout_as_transient_error:, 41	step_configs/prepare_fp_config/continue_on_num_success
step_configs/prepare_explore_config (Argument)	continue_on_num_success:, 33
prep_explore_config:, 25	step_configs/prepare_fp_config/continue_on_success_ratio
step_configs/prepare_explore_config/continue_on_failed (Argument)	continue_on_success_ratio:, 33
(Argument)	step_configs/prepare_fp_config/executor (Argument)
continue_on_failed:, 27	executor:, 33
step_configs/prepare_explore_config/continue_on_num_success (Argument)	step_configs/prepare_fp_config/executor/type
(Argument)	(Argument)
continue_on_num_success:, 27	type:, 33
step_configs/prepare_explore_config/continue_on_success_ratio (Argument)	step_configs/prepare_fp_config/executor[lebesgue_v2]/extra
(Argument)	extra:, 33
continue_on_success_ratio:, 27	step_configs/prepare_fp_config/executor[lebesgue_v2]/extra/job_type
step_configs/prepare_explore_config/executor (Argument)	job_type:, 34
(Argument)	step_configs/prepare_fp_config/executor[lebesgue_v2]/extra/program_id
executor:, 27	program_id:, 34
step_configs/prepare_explore_config/executor/type	step_configs/prepare_fp_config/executor[lebesgue_v2]/extra/scass_type
(Argument)	scass_type:, 34
type:, 27	step_configs/prepare_fp_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
step_configs/prepare_explore_config/executor[lebesgue_v2] (Argument)	template_cover_cmd_escape_bug:, 34
(Argument)	step_configs/prepare_fp_config/parallelism
extra:, 27	parallelism:, 33
step_configs/prepare_explore_config/executor[lebesgue_v2] (Argument)	step_configs/prepare_fp_config/template_config
(Argument)	(Argument)
job_type:, 28	template_config:, 31
step_configs/prepare_explore_config/executor[lebesgue_v2] (Argument)	step_configs/prepare_fp_config/template_config/envs
(Argument)	
program_id:, 28	
step_configs/prepare_explore_config/executor[lebesgue_v2] (Argument)	
(Argument)	
scass_type:, 28	
step_configs/prepare_explore_config/executor[lebesgue_v2] (Argument)	
(Argument)	
template_cover_cmd_escape_bug:, 28	
step_configs/prepare_explore_config/parallelism	
(Argument)	
parallelism:, 27	
step_configs/prepare_explore_config/template_config	
(Argument)	
template_config:, 26	

```

    (Argument)
    envs:, 32
step_configs/prep_fp_config/template_config/image    (Argument)
    (Argument)
    image:, 20
step_configs/prep_fp_config/template_config/retry_on_transient_error
    (Argument)
    retry_on_transient_error:, 20
step_configs/prep_fp_config/template_config/timeout (Argument)
    (Argument)
    timeout:, 20
step_configs/prep_fp_config/template_config/timeout_as_transient_error
    (Argument)
    timeout_as_transient_error:, 21
step_configs/prep_train_config (Argument)
    prep_train_config:, 20
step_configs/prep_train_config/continue_on_failed (Argument)
    (Argument)
    continue_on_failed:, 21
step_configs/prep_train_config/continue_on_num_success (Argument)
    (Argument)
    continue_on_num_success:, 21
step_configs/prep_train_config/continue_on_success_ratio (Argument)
    (Argument)
    continue_on_success_ratio:, 21
step_configs/prep_train_config/executor
    (Argument)
    executor:, 21
step_configs/prep_train_config/executor/type
    (Argument)
    type:, 21
step_configs/prep_train_config/executor[lebesgue_v2]/extra
    (Argument)
    extra:, 22
step_configs/prep_train_config/executor[lebesgue_v2]/extra/job_type
    (Argument)
    job_type:, 22
step_configs/prep_train_config/executor[lebesgue_v2]/extra/program_id
    (Argument)
    program_id:, 22
step_configs/prep_train_config/executor[lebesgue_v2]/extra/scass_type
    (Argument)
    scass_type:, 22
step_configs/prep_train_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
    (Argument)
    template_cover_cmd_escape_bug:, 22
step_configs/prep_train_config/parallelism
    (Argument)
    parallelism:, 21
step_configs/prep_train_config/template_config
    (Argument)
    template_config:, 20
step_configs/prep_train_config/template_config/envs (Argument)
    (Argument)
    envs:, 21
step_configs/prep_train_config/template_config/image
    (Argument)
    image:, 20
step_configs/prep_train_config/template_config/retry_on_transient_error
    (Argument)
    retry_on_transient_error:, 20
step_configs/prep_train_config/template_config/timeout
    (Argument)
    timeout:, 20
step_configs/prep_train_config/template_config/timeout_as_transient_error
    (Argument)
    timeout_as_transient_error:, 21
step_configs/run_explore_config (Argument)
    run_explore_config:, 28
step_configs/run_explore_config/continue_on_failed
    (Argument)
    continue_on_failed:, 29
step_configs/run_explore_config/continue_on_num_success
    (Argument)
    continue_on_num_success:, 30
step_configs/run_explore_config/continue_on_success_ratio
    (Argument)
    continue_on_success_ratio:, 30
step_configs/run_explore_config/executor (Argument)
    (Argument)
    executor:, 30
step_configs/run_explore_config/executor/type
    (Argument)
    type:, 30
step_configs/run_explore_config/executor[lebesgue_v2]/extra
    (Argument)
    extra:, 30
step_configs/run_explore_config/executor[lebesgue_v2]/extra/job_type
    (Argument)
    job_type:, 31
step_configs/run_explore_config/executor[lebesgue_v2]/extra/program_id
    (Argument)
    program_id:, 31
step_configs/run_explore_config/executor[lebesgue_v2]/extra/scass_type
    (Argument)
    scass_type:, 31
step_configs/run_explore_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
    (Argument)
    template_cover_cmd_escape_bug:, 31
step_configs/run_explore_config/parallelism
    (Argument)
    parallelism:, 30
step_configs/run_explore_config/template_config
    (Argument)
    template_config:, 29
step_configs/run_explore_config/template_config/envs
    (Argument)
    envs:, 29

```



```

step_configs/run_explore_config/template_config/image (Argument)
    (Argument) image:, 35
    image:, 29 step_configs/run_fp_config/template_config/retry_on_transient_error
step_configs/run_explore_config/template_config/retry_on_transient_error
    (Argument) retry_on_transient_error:, 35
    retry_on_transient_error:, 29 step_configs/run_fp_config/template_config/timeout
step_configs/run_explore_config/template_config/timeout (Argument)
    (Argument) timeout:, 35
    timeout:, 29 step_configs/run_fp_config/template_config/timeout_as_transient_error
step_configs/run_explore_config/template_config/timeout_as_transient_error
    (Argument) timeout_as_transient_error:, 35
    timeout_as_transient_error:, 29 step_configs/run_train_config (Argument)
step_configs/run_fp_config (Argument) run_train_config:, 22
    run_fp_config:, 34 step_configs/run_train_config/continue_on_failed
step_configs/run_fp_config/continue_on_failed (Argument)
    (Argument) continue_on_failed:, 24
    continue_on_failed:, 35 step_configs/run_train_config/continue_on_num_success
step_configs/run_fp_config/continue_on_num_success (Argument)
    (Argument) continue_on_num_success:, 24
    continue_on_num_success:, 35 step_configs/run_train_config/continue_on_success_ratio
step_configs/run_fp_config/continue_on_success_ratio (Argument)
    (Argument) continue_on_success_ratio:, 24
    continue_on_success_ratio:, 36 step_configs/run_train_config/executor (Argument)
step_configs/run_fp_config/executor (Argument)
    (Argument) executor:, 24
    executor:, 36 step_configs/run_train_config/executor/type
step_configs/run_fp_config/executor/type (Argument)
    (Argument) type:, 24
    type:, 36 step_configs/run_train_config/executor[lebesgue_v2]/extra
step_configs/run_fp_config/executor[lebesgue_v2]/extra (Argument)
    (Argument) extra:, 24
    extra:, 36 step_configs/run_train_config/executor[lebesgue_v2]/extra/job_type
step_configs/run_fp_config/executor[lebesgue_v2]/extra/job_type
    (Argument) job_type:, 25
    job_type:, 37 step_configs/run_train_config/executor[lebesgue_v2]/extra/program_id
step_configs/run_fp_config/executor[lebesgue_v2]/extra/program_id
    (Argument) program_id:, 25
    program_id:, 37 step_configs/run_train_config/executor[lebesgue_v2]/extra/scass_type
step_configs/run_fp_config/executor[lebesgue_v2]/extra/scass_type
    (Argument) scass_type:, 25
    scass_type:, 36 step_configs/run_train_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
step_configs/run_fp_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
    (Argument) template_cover_cmd_escape_bug:, 25
    template_cover_cmd_escape_bug:, 37 step_configs/run_train_config/parallelism
step_configs/run_fp_config/parallelism (Argument)
    (Argument) parallelism:, 24
    parallelism:, 36 step_configs/run_train_config/template_config
step_configs/run_fp_config/template_config (Argument)
    (Argument) template_config:, 23
    template_config:, 34 step_configs/run_train_config/template_config/envs
step_configs/run_fp_config/template_config/envs (Argument)
    (Argument) envs:, 23
    envs:, 35 step_configs/run_train_config/template_config/image
step_configs/run_fp_config/template_config/image (Argument)
    (Argument)

```

```

    image:, 23
step_configs/run_train_config/template_config/retry_on_transient_error
    (Argument)
    retry_on_transient_error:, 38
step_configs/run_train_config/template_config/timeout (Argument)
    (Argument)
    timeout:, 38
step_configs/run_train_config/template_config/timeout_as_transient_error
    (Argument)
    timeout_as_transient_error:, 38
step_configs/select_confs_config (Argument)
    select_confs_config:, 37
step_configs/select_confs_config/continue_on_failed (Argument)
    continue_on_failed:, 38
step_configs/select_confs_config/continue_on_num_success (Argument)
    continue_on_num_success:, 38
step_configs/select_confs_config/continue_on_success_ratio
    (Argument)
    continue_on_success_ratio:, 39
step_configs/select_confs_config/executor
    (Argument)
    executor:, 39
step_configs/select_confs_config/executor/type
    (Argument)
    type:, 39
step_configs/select_confs_config/executor[lebesgue_v2]/extra
    (Argument)
    extra:, 39
step_configs/select_confs_config/executor[lebesgue_v2]/extra/job_type
    (Argument)
    job_type:, 40
step_configs/select_confs_config/executor[lebesgue_v2]/extra/program_id
    (Argument)
    program_id:, 40
step_configs/select_confs_config/executor[lebesgue_v2]/extra/scass_type
    (Argument)
    scass_type:, 39
step_configs/select_confs_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
    (Argument)
    template_cover_cmd_escape_bug:, 40
step_configs/select_confs_config/parallelism
    (Argument)
    parallelism:, 39
step_configs/select_confs_config/template_config
    (Argument)
    template_config:, 37
step_configs/select_confs_config/template_config/envs
    (Argument)
    envs:, 38
step_configs/select_confs_config/template_config/image
    (Argument)
    image:, 37
step_configs/select_confs_config/template_config/retry_on_transient_error
    (Argument)
    retry_on_transient_error:, 38
step_configs/select_confs_config/template_config/timeout
    (Argument)
    timeout:, 38
step_configs/select_confs_config/template_config/timeout_as_transient_error
    (Argument)
    timeout_as_transient_error:, 38
step_configs:
    step_configs (Argument), 20
submit_args() (in module dp-
    gen2.entrypoint.submit_args), 76
submit_concurrent_learning() (in module dp-
    gen2.entrypoint.submit), 76
submit_step_keys() (in module dp-
    gen2.entrypoint.submit), 76
task_list(dpgen2.exploration.task.task.ExplorationTaskGroup
    property), 89
task_list(dpgen2.exploration.task.task.FooTaskGroup
    property), 89
task_max:
    fp[vasp]/task_max (Argument), 15
template_conf_args() (in module dp-
    gen2.utils.step_config), 110
template_config (Argument)
    template_config:, 56
template_config/envs (Argument)
    envs:, 57
template_config/image (Argument)
    image:, 56
template_config/retry_on_transient_error (Ar-
    gument)
    retry_on_transient_error:, 56
template_config/scass_type (Argument)
    timeout:, 56
template_config/timeout_as_transient_error
    (Argument)
    timeout_as_transient_error:, 57
template_config:
    default_step_config/template_config
        (Argument), 46
    step_configs/cl_step_config/template_config
        (Argument), 43
    step_configs/collect_data_config/template_config
        (Argument), 40
    step_configs/prep_explore_config/template_config
        (Argument), 26
    step_configs/prep_fp_config/template_config
        (Argument), 31
    step_configs/prep_train_config/template_config
        (Argument), 20

```

```

step_configs/run_explore_config/template_config (Argument), 38
(Argument), 29
step_configs/run_fp_config/template_config timeout_as_transient_error:
(Argument), 34
step_configs/run_train_config/template_config (Argument), 47
(Argument), 23
step_configs/select_confs_config/template_config (Argument), 44
(Argument), 37
template_config (Argument), 56
template_cover_cmd_escape_bug:
default_step_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 48
executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug (Argument), 32
(Argument), 56
step_configs/cl_step_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 46
step_configs/collect_data_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 43
step_configs/prep_explore_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 28
step_configs/prep_fp_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 34
step_configs/prep_train_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 22
step_configs/run_explore_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 31
step_configs/run_fp_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 37
step_configs/run_train_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 25
step_configs/select_confs_config/executor[lebesgue_v2]/extra/template_cover_cmd_escape_bug
(Argument), 40
template_script:
train[dp]/template_script (Argument), 19
timeout:
default_step_config/template_config/timeout
(Argument), 46
step_configs/cl_step_config/template_config/timeout
(Argument), 44
step_configs/collect_data_config/template_config/timeout
(Argument), 41
step_configs/prep_explore_config/template_config/timeout
(Argument), 26
step_configs/prep_fp_config/template_config/timeout
(Argument), 32
step_configs/prep_train_config/template_config/timeout
(Argument), 20
step_configs/run_explore_config/template_config/timeout
(Argument), 29
step_configs/run_fp_config/template_config/timeout
(Argument), 35
step_configs/run_train_config/template_config/timeout
(Argument), 23
step_configs/select_confs_config/template_config/timeout
(Argument), 44
template_config/timeout (Argument), 56
template_config/timeout_as_transient_error
(Argument), 41
step_configs/prep_explore_config/template_config/timeout
(Argument), 26
step_configs/prep_fp_config/template_config/timeout_as_transient_error
(Argument), 32
step_configs/prep_train_config/template_config/timeout
(Argument), 21
step_configs/run_explore_config/template_config/timeout
(Argument), 29
step_configs/run_fp_config/template_config/timeout_as_transient_error
(Argument), 35
step_configs/run_train_config/template_config/timeout
(Argument), 23
step_configs/select_confs_config/template_config/timeout
(Argument), 44
template_config/timeout_as_transient_error
(Argument), 56
to_str() (dpngen2.op.md_settings.MDSettings method),
99
train (Argument)
train/extra/template_cover_cmd_escape_bug
train/type (Argument)
train:
train (Argument), 17
training_args() (dpngen2.op.run_dp_train.RunDPTrain static
method), 99
train[dp]/config (Argument)
train[dp]/config/init_model_num_steps (Argument), 18
train[dp]/config/init_model_old_ratio (Argument), 18
train[dp]/config/init_model_policy (Argument), 18
train[dp]/config/init_model_start_lr (Argument), 18
train[dp]/config/init_model_start_pref_e (Argument), 18
train[dp]/config/init_model_start_pref_f (Argument), 18

```



init\_model\_start\_pref\_f:, 19  
 train[dp]/config/init\_model\_start\_pref\_v (Argument)  
 init\_model\_start\_pref\_v:, 19  
 train[dp]/numb\_models (Argument)  
 numb\_models:, 19  
 train[dp]/template\_script (Argument)  
 template\_script:, 19  
 TrajsExplorationReport (class in dp-gen2.exploration.report.trajs\_report), 78  
 TrustLevel (class in dp-gen2.exploration.selector.trust\_level), 85  
 type:  
   default\_step\_config/executor/type (Argument), 48  
   executor/type (Argument), 55  
   explore/type (Argument), 16  
   fp/type (Argument), 15  
   step\_configs/cl\_step\_config/executor/type (Argument), 45  
   step\_configs/collect\_data\_config/executor/type (Argument), 42  
   step\_configs/prep\_explore\_config/executor/type (Argument), 27  
   step\_configs/prep\_fp\_config/executor/type (Argument), 33  
   step\_configs/prep\_train\_config/executor/type (Argument), 21  
   step\_configs/run\_explore\_config/executor/type (Argument), 30  
   step\_configs/run\_fp\_config/executor/type (Argument), 36  
   step\_configs/run\_train\_config/executor/type (Argument), 24  
   step\_configs/select\_confs\_config/executor/type (Argument), 39  
   train/type (Argument), 18  
 type\_map (Argument)  
   type\_map:, 53  
 type\_map:  
   inputs/type\_map (Argument), 19  
   type\_map (Argument), 53

v\_trust\_lo:  
   explore[lmp]/v\_trust\_lo (Argument), 17  
 variant\_executor() (in module dp-gen2.utils.step\_config), 111  
 variant\_explore() (in module dp-gen2.entrypoint.submit\_args), 76  
 variant\_fp() (in module dp-gen2.entrypoint.submit\_args), 76  
 variant\_train() (in module dp-gen2.entrypoint.submit\_args), 76  
 vasp\_args() (dp-gen2.op.run\_vasp.RunVasp static method), 101  
 vasp\_args() (in module dp-gen2.entrypoint.submit\_args), 76  
 VaspInputs (class in dp-gen2.fp.vasp), 93

## W

watch() (in module dp-gen2.entrypoint.watch), 77  
 wf\_global\_workflow() (in module dp-gen2.entrypoint.submit), 76  
 workflow\_concurrent\_learning() (in module dp-gen2.entrypoint.submit), 76  
 workflow\_config\_from\_dict() (in module dp-gen2.utils.dflow\_config), 109  
 write\_data\_to\_input\_script() (dp-gen2.op.run\_dp\_train.RunDPTrain static method), 99  
 write\_other\_to\_input\_script() (dp-gen2.op.run\_dp\_train.RunDPTrain static method), 99

## U

update\_finished\_steps() (in module dp-gen2.entrypoint.watch), 77  
 upload\_python\_package (Argument)  
   upload\_python\_package:, 20  
 upload\_python\_package:  
   upload\_python\_package (Argument), 20

## V

v\_trust\_hi:  
   explore[lmp]/v\_trust\_hi (Argument), 17