Hands on Auto-test on Aluminum

1. Three operations

There are three operations in auto-test package, namely `make` , `run` , and `post`

1.1 Make

The `INCAR` , `POSCAR` , `POTCAR` input files for VASP or `in.lammps` , `conf.lmp` , and the interatomic potential files for LAMMPS will be generated in the directory `confs/std-*/relaxation/relax_task` for relaxation or `confs/std-*/eos_00/task.[0-9]*[0-9]` e.g. for EOS. The `machine.json` file is not needed for `make` . Example:

```
dpgen autotest make PARAM
```

1.2 Run

The jobs would be dispatched according to the parameter in `MACHINE` file and the calculation results would be sent back. Example:

```
dpgen autotest run PARAM MACHINE
```

1.3 Post

The post process of calculation results would be performed. `result.json` in json format in `confs/mp-*/relaxation/relax_task` for relaxation and `result.json` in json format and `result.out` in txt format in `confs/mp-*/eos_00` e.g. for EOS will be generated. In addition, `result_task.json` in json format would also be generated in each `confs/std-*/eos_00/task.[0-9]*[0-9]` .The `MACHINE` file is also not needed for `post` . Example:

```
dpgen autotest post PARAM
```

2. Structure relaxation

All the property tests should be based on the equilibrium state calculated either by `VASP` or `LAMMPS` . The structure after relaxation is supposed to exist as the file like `confs/mp-*/relaxation/relax_task/CONTCAR` and the further property tests would normally start from this configuration.

2.1 Input example

2.1.1 An example of the input file for relaxation by VASP:

```
{
    "structures":            ["confs/std-*"],
    "interaction": {
            "type":           "vasp",
            "incar":          "vasp_input/INCAR",
            "potcar_prefix":  "vasp_input",
            "potcars":       {"Al": "POTCAR.al"}
        },
    "relaxation": {
            "cal_type":       "relaxation",
            "cal_setting":   {"relax_pos":       true,
                              "relax_shape":     true,
                              "relax_vol":       true,
                              "ediff":           1e-6,
                              "ediffg":         -0.01,
                              "encut":           650,
                              "kspacing":        0.1,
                              "kgamma":          false}
        }
}
```

For VASP relaxation and all the property calculations, **the initial INCAR file must be given by user** and the package would change the `ISIF` and `NSW` parameter according to the property type. Besides, users can also set the `cal_setting` dictionary in the `relaxation` part to make the final changes on INCAR.

Key words	data structure	example	description
structures	List of String	["confs/std-*"]	path of different structures
interaction	Dict	See above	description of the task type and atomic interaction
type	String	"vasp"	task type
incar	String	"vasp/input/INCAR"	the path for INCAR file in vasp
potcar_prefix	String	"vasp_input"	the prefix of path for POTCAR file in vasp, default = ""
potcars	Dict	{"Al": "POTCAR.al"}	key is element type and value is potcar name

Key words	data structure	example	description
relaxation	Dict	See above	the calculation type and setting for relaxation
cal_type	String	"relaxation" or "static"	calculation type
cal_setting	Dict	See above	calculation setting
relax_pos	Boolean	True	relax atomic position or not, default = True for relaxation
relax_shape	Boolean	True	relax box shape or not, default = True for relaxation
relax_vol	Boolean	True	relax box volume or not, default = True for relaxation
ediff	Float	1e-6	set `EDIFF` parameter in INCAR files
ediffg	Float	-0.01	set `EDIFFG` parameter in INCAR files
encut	Int	650	set `encut` parameter in INCAR files
kspacing	Float	0.1	set `KSPACING` parameter in INCAR files
kgamma	Boolean	False	set `KGAMMA` parameter in INCAR files

2.1.2 An example of the input file for relaxation by LAMMPS:

```
{
    "structures":         ["confs/std-*"],
    "interaction": {
            "type":        "deepmd",
            "model":       "frozen_model.pb",
            "in_lammps":   "lammps_input/in.lammps",
            "type_map":    {"Al": 0}
        },
    "relaxation": {
            "cal_setting":{"etol": 1e-12,
                           "ftol": 1e-6,
                           "maxiter": 5000,
                           "maximal": 500000}
        }
}
```

Other key words different from vasp:

Key words	data structure	example	description
model	String or List of String	"frozen_model.pb"	model file for atomic interaction
in_lammps	String	"lammps_input/in.lammps"	input file for lammps commands
type_map	Dict	{"Al": 0}	key is element type and value is type number. DP starts from 0, others starts from 1
etol	Float	1e-12	stopping tolerance for energy
ftol	Float	1e-6	stopping tolerance for force
maxiter	Int	5000	max iterations of minimizer
maxeval	Int	500000	max number of force/energy evaluations

For LAMMPS relaxation and all the property calculations, **package will help to generate `in.lammps` file for user automatically** according to the property type. We can also make the final changes in the `minimize` setting (`minimize etol ftol maxiter maxeval`) in `in.lammps` . In addition, users can apply the input file for lammps commands in the `interaction` part. For further information of the LAMMPS relaxation, we refer users to minimize command.

2.2 Relaxation: make

The list of the directories storing structures are `["confs/std-*"]` in this example. For single element system, if `POSCAR` doesn't exist in the directories: `std-fcc` , `std-hcp` , `std-dhcp` , `std-bcc` , `std-diamond` , and `std-sc` , the package will automatically generate the standard crystal structures **fcc**, **hcp**, **dhcp**, **bcc**, **diamond**, and **sc** in the corresponding directories, respectively. In other conditions, the package would try to dowload structure from Materials Project through API.

2.2.1 VASP relaxation

```
dpgen autotest make relaxation.json
tree confs/std-fcc/relaxation/
```

the output would be:

```
confs/std-fcc/relaxation/
|-- INCAR
|-- POTCAR
`-- relax_task
    |-- CONTCAR
    |-- INCAR -> ../INCAR
    |-- inter.json
    |-- KPOINTS
    |-- OUTCAR
    |-- POSCAR -> ../../POSCAR
    |-- POTCAR -> ../POTCAR
    |-- result.json
    `-- task.json
```

`inter.json` records the information in the `interaction` dictionary and `task.json` records the information in the `relaxation` dictionary.

2.2.2 LAMMPS relaxation

```
ssh root@IP
conda activate ali-dpgen
cd /workshop/deepmd
dpgen autotest make relaxation.json
tree confs/std-fcc/
```

the output would be:

```
confs/std-fcc/
|-- POSCAR
`-- relaxation
    |-- frozen_model.pb -> ../../../frozen_model.pb
    |-- in.lammps
    `-- relax_task
        |-- conf.lmp
        |-- frozen_model.pb -> ../frozen_model.pb
        |-- in.lammps -> ../in.lammps
        |-- inter.json
        |-- POSCAR -> ../../POSCAR
        `-- task.json
```

the `conf.lmp` is the input configuration and `in.lammps` is the input command file for lammps.

2.3 Relaxation: run

The `machine.json` file should be applied in this process and the machine parameters (eg. GPU or CPU) are determined according to the task type (VASP or LAMMPS). Then in each work path, the corresponding tasks would be submitted and the results would be sent back through make_dispatcher.

Take `deepmd` run for example:

```
nohup dpgen autotest run relaxation.json machine-ali.json > run.result 2>&1 &
tree confs/std-fcc/relaxation/
```

the output would be:

```
confs/std-fcc/relaxation/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- jr.json
`-- relax_task
    |-- conf.lmp
    |-- dump.relax
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- log.lammps
    |-- outlog
    |-- POSCAR -> ../../POSCAR
    `-- task.json
```

`dump.relax` is the file storing configurations and `log.lammps` is the output file for lammps.

2.4 Relaxation: post

Take `deepmd` post for example:

```
dpgen autotest post relaxation.json
tree confs/std-fcc/relaxation/
```

```
confs/std-fcc/relaxation/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- jr.json
`-- relax_task
    |-- conf.lmp
    |-- CONTCAR
    |-- dump.relax
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- log.lammps
    |-- outlog
    |-- POSCAR -> ../../POSCAR
    |-- result.json
    `-- task.json
```

`result.json` stores the box cell, coordinates, energy, force, virial,... information of each frame in the relaxation trajectory and `CONTCAR` is the final equilibrium configuration.

result.json：

```
{
    "@module": "dpdata.system",
    "@class": "LabeledSystem",
    "data": {
        "atom_numbs": [
            1
        ],
        "atom_names": [
            "Al"
        ],
        "atom_types": {
            "@module": "numpy",
            "@class": "array",
            "dtype": "int64",
            "data": [
                0
            ]
        },
        "orig": {
            "@module": "numpy",
            "@class": "array",
            "dtype": "int64",
            "data": [
                0,
                0,
                0
            ]
        },
        "cells": {
            "@module": "numpy",
            "@class": "array",
            "dtype": "float64",
            "data": [
                [
                    [
                        2.8637824638,
                        0.0,
                        0.0
                    ],
                    [
                        1.4318912319,
                        2.4801083646,
                        0.0
                    ],
                    [
                        1.4318912319,
                        0.8267027882,
                        2.3382685902
                    ]
                ],
                [
                    [
                        2.8549207998018438,
                        0.0,
                        0.0
```

```
                ],
                [
                    1.4274603999009239,
                    2.472433938457684,
                    0.0
                ],
                [
                    1.4274603999009212,
                    0.8241446461525599,
                    2.331033071844216
                ]
            ],
            [
                [
                    2.854920788303194,
                    0.0,
                    0.0
                ],
                [
                    1.427460394144466,
                    2.472433928487206,
                    0.0
                ],
                [
                    1.427460394154763,
                    0.8241446428350139,
                    2.331033062460779
                ]
            ]
        ]
    },
    "coords": {
        "@module": "numpy",
        "@class": "array",
        "dtype": "float64",
        "data": [
            [
                [
                    0.0,
                    0.0,
                    0.0
                ]
            ],
            [
                [
                    5.709841595683707e-25,
                    -4.3367974740910857e-19,
                    0.0
                ]
            ],
            [
                [
                    -8.673606219968035e-19,
                    8.673619637565944e-19,
                    8.673610853102186e-19
```

```
                ]
            ]
        ]
    },
    "energies": {
        "@module": "numpy",
        "@class": "array",
        "dtype": "float64",
        "data": [
            -3.745029,
            -3.7453815,
            -3.7453815
        ]
    },
    "forces": {
        "@module": "numpy",
        "@class": "array",
        "dtype": "float64",
        "data": [
            [
                [
                    0.0,
                    -6.93889e-18,
                    -3.46945e-18
                ]
            ],
            [
                [
                    1.38778e-17,
                    6.93889e-18,
                    -1.73472e-17
                ]
            ],
            [
                [
                    1.38778e-17,
                    1.73472e-17,
                    -4.51028e-17
                ]
            ]
        ]
    },
    "virials": {
        "@module": "numpy",
        "@class": "array",
        "dtype": "float64",
        "data": [
            [
                [
                    -0.07534992071654338,
                    1.2156615579052586e-17,
                    1.3904892126132796e-17
                ],
                [
                    1.2156615579052586e-17,
```

```
                    -0.07534992071654338,
                    4.61571024026576e-12
                ],
                [
                    1.3904892126132796e-17,
                    4.61571024026576e-12,
                    -0.07534992071654338
                ]
            ],
            [
                [
                    -9.978994290457664e-08,
                    -3.396452753975288e-15,
                    8.785831629151552e-16
                ],
                [
                    -3.396452753975288e-15,
                    -9.991375413666671e-08,
                    5.4790751628409565e-12
                ],
                [
                    8.785831629151552e-16,
                    5.4790751628409565e-12,
                    -9.973497959053003e-08
                ]
            ],
            [
                [
                    1.506940521266962e-11,
                    1.1152016233536118e-11,
                    -8.231900529157644e-12
                ],
                [
                    1.1152016233536118e-11,
                    -6.517665029355618e-11,
                    -6.33706710415926e-12
                ],
                [
                    -8.231900529157644e-12,
                    -6.33706710415926e-12,
                    5.0011471096530724e-11
                ]
            ]
        ]
    },
    "stress": {
        "@module": "numpy",
        "@class": "array",
        "dtype": "float64",
        "data": [
            [
                [
                    -7.2692250000000005,
                    1.1727839e-15,
                    1.3414452e-15
```

```
                    ],
                    [
                        1.1727839e-15,
                        -7.2692250000000005,
                        4.4529093000000003e-10
                    ],
                    [
                        1.3414452e-15,
                        4.4529093000000003e-10,
                        -7.2692250000000005
                    ]
                ],
                [
                    [
                        -9.71695e-06,
                        -3.3072633e-13,
                        8.5551193e-14
                    ],
                    [
                        -3.3072633e-13,
                        -9.729006000000001e-06,
                        5.3351969e-10
                    ],
                    [
                        8.5551193e-14,
                        5.3351969e-10,
                        -9.711598e-06
                    ]
                ],
                [
                    [
                        1.4673689e-09,
                        1.0859169e-09,
                        -8.0157343e-10
                    ],
                    [
                        1.0859169e-09,
                        -6.3465139e-09,
                        -6.1706584e-10
                    ],
                    [
                        -8.0157343e-10,
                        -6.1706584e-10,
                        4.8698191e-09
                    ]
                ]
            ]
        }
    }
}
```


3. property

3.1 Input example

Here we take deepmd for example and the input file for other task types is similar.

```
{
    "structures":       ["confs/std-*"],
    "interaction": {
        "type":          "deepmd",
        "model":         "frozen_model.pb",
        "deepmd_version":"1.2.0",
        "type_map":     {"Al": 0}
    },
    "properties": [
        {
         "type":         "eos",
         "vol_start":    0.9,
         "vol_end":      1.1,
         "vol_step":     0.01
        },
        {
         "type":         "elastic",
         "norm_deform":  2e-2,
         "shear_deform": 5e-2
        },
        {
         "type":             "vacancy",
         "supercell":        [3, 3, 3],
         "start_confs_path": "../vasp/confs"
        },
        {
         "type":         "interstitial",
         "supercell":    [3, 3, 3],
         "insert_ele":   ["Al"],
         "conf_filters":{"min_dist": 1.5},
         "cal_setting": {"input_prop": "lammps_input/lammps_high"}
        },
        {
         "type":           "surface",
         "min_slab_size":   10,
         "min_vacuum_size":11,
         "max_miller":      2,
         "cal_type":       "static"
        }
        ]
}
```

Universal key words for properties

Key words	data structure	example	description

Key words	data structure	example	description
type	String	"eos"	specifying the property type
skip	Boolean	true	whether to skip current property or not
start_confs_path	String	"../vasp/confs"	starting from the equilibrium configuration in other path only for the current property type
cal_setting["input_prop"]	String	"lammps_input/lammps_high"	input commands file for lammps
cal_setting["overwrite_interaction"]	Dict		overwrite the interaction in the `interaction` part only for the current property type

other parameters in `cal_setting` and `cal_type` in `relaxation` also apply in `property` .

Key words for **EOS**

Key words	data structure	example	description
vol_start	Float	0.9	the starting volume related to the equilibrium structure

Key words	data structure	example	description
vol_end	Float	1.1	the biggest volume related to the equilibrium structure
vol_step	Float	0.01	the volume increment related to the equilibrium structure

Key words for **Elastic**

Key words	data structure	example	description
norm_deform	Float	2e-2	specifying the deformation in xx, yy, zz, default = 2e-3
shear_deform	Float	5e-2	specifying the deformation in other directions, default = 5e-3

Key words for **Vacancy**

Key words	data structure	example	description
supercell	Lisf of Int	[3,3,3]	the supercell to be constructed, default = [1,1,1]

Key words for **Interstitial**

Key words	data structure	example	description
insert_ele	Lisf of String	["Al"]	the element to be inserted
supercell	Lisf of Int	[3,3,3]	the supercell to be constructed, default = [1,1,1]
conf_filters	Dict	"min_dist": 1.5	filter out the undesirable configuration

Key words for **Surface**

Key words	data structure	example	description
min_slab_size	Int	10	minimum size of slab thickness
min_vacuum_size	Int	11	minimum size of vacuume width

Key words	data structure	example	description
pert_xz	Float	0.01	perturbation through xz direction used to compute surface energy, default = 0.01
max_miller	Int	2	the maximum miller index

3.2 Property: make

```
dpgen autotest make property.json
```

EOS output:

```
confs/std-fcc/eos_00/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- task.000000
|   |-- conf.lmp
|   |-- eos.json
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- POSCAR.orig -> ../../relaxation/relax_task/CONTCAR
|   `-- task.json
|-- task.000001
|   |-- conf.lmp
|   |-- eos.json
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- POSCAR.orig -> ../../relaxation/relax_task/CONTCAR
|   `-- task.json
...
`-- task.000019
    |-- conf.lmp
    |-- eos.json
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps
    |-- inter.json
    |-- POSCAR
    |-- POSCAR.orig -> ../../relaxation/relax_task/CONTCAR
    `-- task.json
```

`eos.json` records the `volume` and `scale` of the corresponding task.

Elastic output:

```
confs/std-fcc/elastic_00/
|-- equi.stress.json
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- POSCAR -> ../relaxation/relax_task/CONTCAR
|-- task.000000
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- strain.json
|   `-- task.json
|-- task.000001
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- strain.json
|   `-- task.json
...
`-- task.000023
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR
    |-- strain.json
    `-- task.json
```

`strain.json` records the deformation information of the corresponding task.

Vacancy output:

```
confs/std-fcc/vacancy_00/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- POSCAR -> ../relaxation/relax_task/CONTCAR
`-- task.000000
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR
    |-- supercell.json
    `-- task.json
```

`supercell.json` records the supercell size information of the corresponding task.

Interstitial output:

```
confs/std-fcc/interstitial_00/
|-- element.out
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- POSCAR -> ../relaxation/relax_task/CONTCAR
|-- task.000000
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   |-- supercell.json
|   `-- task.json
`-- task.000001
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR
    |-- supercell.json
    `-- task.json
```

`supercell.json` records the supercell size information of the corresponding task and `element.out` records the inserted element type of each task.

Surface output:

```
confs/std-fcc/surface_00/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- POSCAR -> ../relaxation/relax_task/CONTCAR
|-- task.000000
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- miller.json
|   |-- POSCAR
|   |-- POSCAR.tmp
|   `-- task.json
|-- task.000001
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- miller.json
|   |-- POSCAR
|   |-- POSCAR.tmp
|   `-- task.json
...
`-- task.000008
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- miller.json
    |-- POSCAR
    |-- POSCAR.tmp
    `-- task.json
```

`miller.json` records the miller index of the corresponding task.

3.3 Property: run

```
nohup dpgen autotest run property.json machine-ali.json > run.result 2>&1 &
```

the result file `log.lammps` , `dump.relax` , and `outlog` would be sent back.

3.4 Property: post

```
dpgen autotest post property.json
```

EOS output:

`reult.out:`

```
conf_dir: /root/auto_test_example/0703_rehearsal/deepmd/confs/std-fcc/eos_00
 VpA(A^3)  EpA(eV)
 14.808   -3.7194
 14.973   -3.7242
 15.138   -3.7285
 15.302   -3.7323
 15.467   -3.7356
 15.631   -3.7385
 15.796   -3.7409
 15.960   -3.7428
 16.125   -3.7442
 16.289   -3.7451
 16.454   -3.7454
 16.618   -3.7451
 16.783   -3.7440
 16.947   -3.7423
 17.112   -3.7396
 17.277   -3.7360
 17.441   -3.7314
 17.606   -3.7254
 17.770   -3.7180
 17.935   -3.7088
```

result.json:


```
{
   "14.808453313267595": -3.7194474,
   "14.972991683415014": -3.7242038,
   "15.13753005356243": -3.7284845,
   "15.30206842370985": -3.7322877,
   "15.466606793857267": -3.7356189,
   "15.631145164004685": -3.7384827,
   "15.7956835341521": -3.7408759,
   "15.96022190429952": -3.7427885,
   "16.12476027444694": -3.7441995,
   "16.289298644594354": -3.7450777,
   "16.453837014741772": -3.7453815,
   "16.61837538488919": -3.7450585,
   "16.782913755036606": -3.7440445,
   "16.947452125184025": -3.7422635,
   "17.111990495331444": -3.7396287,
   "17.276528865478863": -3.736038,
   "17.441067235626278": -3.7313635,
   "17.605605605773697": -3.7254247,
   "17.770143975921115": -3.7179689,
   "17.934682346068534": -3.7087655
}
```


Elastic output:

result.out:

```
 /root/auto_test_example/0703_rehearsal/deepmd/confs/std-fcc/elastic_00
  134.91   54.33   51.80    3.57   -0.00   -0.00
   54.56  134.60   51.80   -3.54    0.00    0.00
   51.91   51.91  137.02   -0.00    0.00    0.00
    3.88   -3.77   -1.28   35.41    0.00    0.00
   -0.00    0.00    0.00    0.00   35.38    3.86
    0.00    0.00    0.00    0.00    4.03   38.38
 # Bulk   Modulus BV = 80.32 GPa
 # Shear  Modulus GV = 38.41 GPa
 # Youngs Modulus EV = 99.38 GPa
 # Poission Ratio uV = 0.29
```

result.json:

```
{
    "elastic_tensor": [
        134.90955999999997,
        54.329958699999985,
        51.802386099999985,
        3.5745279599999993,
        -1.3886325999999648e-05,
        -1.9638233999999486e-05,
        54.55840299999999,
        134.59654699999996,
        51.7972336,
        -3.53972684,
        1.839568799999963e-05,
        8.756799399999951e-05,
        51.91324859999999,
        51.913292199999994,
        137.01763799999998,
        -5.090339399999969e-05,
        6.99251629999996e-05,
        3.736478699999946e-05,
        3.8780564440000007,
        -3.770445632,
        -1.2766205999999956,
        35.41343199999999,
        2.2479590800000023e-05,
        1.3837692000000172e-06,
        -4.959999999495933e-06,
        2.5800000003918792e-06,
        1.4800000030874965e-06,
        2.9000000008417968e-06,
        35.375960199999994,
        3.8608356,
        0.0,
        0.0,
        0.0,
        0.0,
        4.02554856,
        38.375018399999995
    ],
    "BV": 80.3153630222222,
    "GV": 38.40582656,
    "EV": 99.37716395728943,
    "uV": 0.2937771799031088
}
```


Vacancy output:

result.out:

```
/root/auto_test_example/0703_rehearsal/deepmd/confs/std-fcc/vacancy_00
Structure:      Vac_E(eV)  E(eV) equi_E(eV)
[3, 3, 3]-task.000000:   0.735  -96.645 -97.380
```

result.json:


```
{
    "[3, 3, 3]-task.000000": [
        0.7352769999999964,
        -96.644642,
        -97.379919
    ]
}
```


Interstitial output:

result.out:

```
/root/auto_test_example/0703_rehearsal/deepmd/confs/std-fcc/interstitial_00
Insert_ele-Struct: Inter_E(eV)  E(eV) equi_E(eV)
Al-[3, 3, 3]-task.000000:   4.023  -100.848 -104.871
Al-[3, 3, 3]-task.000001:   2.783  -102.088 -104.871
```

result.json:


```
{
    "Al-[3, 3, 3]-task.000000": [
        4.022952000000004,
        -100.84773,
        -104.870682
    ],
    "Al-[3, 3, 3]-task.000001": [
        2.7829520000000088,
        -102.08773,
        -104.870682
    ]
}
```


Surface output:

result.out:

```
/root/auto_test_example/0703_rehearsal/deepmd/confs/std-fcc/surface_00
Miller_Indices:         Surf_E(J/m^2) EpA(eV) equi_EpA(eV)
[1, 1, 1]-task.000000:          0.805      -3.604   -3.745
[2, 2, 1]-task.000001:          0.991      -3.578   -3.745
[1, 1, 0]-task.000002:          0.946      -3.553   -3.745
[2, 2, -1]-task.000003:         0.987      -3.559   -3.745
[2, 1, 1]-task.000004:          1.014      -3.563   -3.745
[2, 1, -1]-task.000005:         1.066      -3.543   -3.745
[2, 1, -2]-task.000006:         1.034      -3.551   -3.745
[2, 0, -1]-task.000007:         0.957      -3.569   -3.745
[2, -1, -1]-task.000008:        0.943      -3.577   -3.745
```

result.json:


```
{
    "[1, 1, 1]-task.000000": [
        0.8051037974207992,
        -3.6035018,
        -3.7453815
    ],
    "[2, 2, 1]-task.000001": [
        0.9913881928811771,
        -3.5781115999999997,
        -3.7453815
    ],
    "[1, 1, 0]-task.000002": [
        0.9457333586026173,
        -3.5529366000000002,
        -3.7453815
    ],
    "[2, 2, -1]-task.000003": [
        0.9868013100872397,
        -3.5590607142857142,
        -3.7453815
    ],
    "[2, 1, 1]-task.000004": [
        1.0138239046484236,
        -3.563035875,
        -3.7453815
    ],
    "[2, 1, -1]-task.000005": [
        1.0661817319108005,
        -3.5432459166666668,
        -3.7453815
    ],
    "[2, 1, -2]-task.000006": [
        1.034003253044026,
        -3.550884125,
        -3.7453815
    ],
    "[2, 0, -1]-task.000007": [
        0.9569958287615818,
        -3.5685403333333334,
        -3.7453815
    ],
    "[2, -1, -1]-task.000008": [
        0.9432935501134583,
        -3.5774615714285716,
        -3.7453815
    ]
}
```


4. Refine and Reproduce

4.1 Refine

Universal for all property tests

4.1.1 input examples

In some cases, we want to refine the calculation results of a property based on previous results by using different convergence criteria like `EDIFF` and `EDIFFG` or higher `ENCUT`. If the parameter of `init_from_suffix` and `output_suffix` are both provided in the input file, `refine` would start based on the results in `init_from_suffix` directory and output the results to `output_suffix` directory. Otherwise, the calculation results would be output to the default suffix `00`. An example of the input file is given below:

```
{
    "structures":       ["confs/std-*"],
    "interaction": {
        "type":          "deepmd",
        "model":         "frozen_model.pb",
        "deepmd_version":"1.2.0",
        "type_map":     {"Al": 0}
    },
    "properties": [
        {
        "type":             "vacancy",
        "init_from_suffix": "00",
        "output_suffix":    "01",
        "cal_setting":     {"input_prop":  "lammps_input/lammps_high"}
        }
        ]
}
```

In this example, `refine` would output the results to `vacancy_01` based on the previous results in `vacancy_00` by using a different input commands file for lammps.

4.1.2 Refine: make

```
dpgen autotest make refine.json
tree confs/std-fcc/vacancy_01/
```

```
confs/std-fcc/vacancy_01/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
`-- task.000000
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR -> ../../vacancy_00/task.000000/CONTCAR
    |-- supercell.json -> ../../vacancy_00/task.000000/supercell.json
    `-- task.json
```

an new directory `vacancy_01` would be established and the starting configuration links to previous results.

4.1.3 Refine: run

```
nohup dpgen autotest run refine.json machine-ali.json > run.result 2>&1 &
```

the run process of `refine` is similar to before.

4.1.4 Refine: post

```
dpgen autotest post refine.json
```

the post process of `refine` is similar to the corresponding property.

4.2 Reproduce

Universal for all property tests except for `elastic`

4.2.1 Input examples

Some times we want to reproduce the initial results with the same configurations for cross validation. This version of auto-test package can accomplish this successfully in all property types except for `Elastic`. An input example for using `deepmd` to reproduce the `VASP` Interstitial results is given as below:

```
{
    "structures":       ["confs/std-*"],
    "interaction": {
        "type":          "deepmd",
        "model":         "frozen_model.pb",
        "deepmd_version":"1.2.0",
        "type_map":     {"Al": 0}
    },
    "properties": [
        {
        "type":             "interstitial",
        "reproduce":        true,
        "init_from_suffix": "00",
        "init_data_path":   "../vasp/confs",
        "reprod_last_frame":       false
        }
        ]
}
```

`reproduce` denotes whether to do `reproduce` or not and the default value is False. `init_data_path` is the path of VASP or LAMMPS initial data to be reproduced. `init_from_suffix` is the suffix of the initial data and the default value is "00". In this case, the VASP Interstitial results are stored in `../vasp/confs/std-*/interstitial_00` and the reproduced Interstitial results would be in `deepmd/confs/std-*/interstitial_reprod`. `reprod_last_frame` denotes if only the last frame is used in reproduce. The default value is True for eos and surface, but is False for vacancy and interstitial.

4.2.2 Reproduce: make

```
dpgen autotest make reproduce.json
tree confs/std-fcc/interstitial_reprod/
```

```
confs/std-fcc/interstitial_reprod/
|-- frozen_model.pb -> ../../../frozen_model.pb
|-- in.lammps
|-- task.000000
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   `-- task.json
|-- task.000001
|   |-- conf.lmp
|   |-- frozen_model.pb -> ../frozen_model.pb
|   |-- in.lammps -> ../in.lammps
|   |-- inter.json
|   |-- POSCAR
|   `-- task.json
...
`-- task.000038
    |-- conf.lmp
    |-- frozen_model.pb -> ../frozen_model.pb
    |-- in.lammps -> ../in.lammps
    |-- inter.json
    |-- POSCAR
    `-- task.json
```

every singe frame in the initial data is split into each task and `in.lammps` would help to do the `static` calculation.

4.2.3 Reproduce: run

```
nohup dpgen autotest run reproduce.json machine-ali.json > run.result 2>&1 &
```

the run process of `reproduce` is similar to before.

4.2.4 Reproduce: post

```
dpgen autotest post reproduce.json
```

output:

```
result.out
```

```
/root/auto_test_example/deepmd/confs/std-fcc/interstitial_reprod
Reproduce: Initial_path Init_E(eV/atom)  Reprod_E(eV/atom)  Difference(eV/atom)
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.020   -3.240   -0.220
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.539   -3.541   -0.002
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.582   -3.582   -0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.582   -3.581    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.594   -3.593    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.594   -3.594    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.598   -3.597    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.600   -3.600    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.600   -3.600    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.601   -3.600    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.602   -3.601    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.603   -3.602    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.603   -3.602    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.603   -3.602    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.603   -3.602    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.603   -3.602    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.603   -3.602    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000000  -3.603   -3.602    0.001
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.345   -3.372   -0.027
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.546   -3.556   -0.009
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.587   -3.593   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.593   -3.599   -0.006
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.600   -3.606   -0.006
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.600   -3.606   -0.006
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.624   -3.631   -0.006
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.634   -3.640   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.637   -3.644   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.637   -3.644   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.638   -3.645   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.638   -3.645   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
.../vasp/confs/std-fcc/interstitial_00/task.000001  -3.639   -3.646   -0.007
```

the comparison of the initial and reproduced results as well as the absolute path of the initial data is recorded.

```
result.json
```

```
{
    "/root/auto_test_example/vasp/confs/std-fcc/interstitial_00/task.000000": {
        "nframes": 18,
        "error": 0.0009738182472213228
    },
    "/root/auto_test_example/vasp/confs/std-fcc/interstitial_00/task.000001": {
        "nframes": 21,
        "error": 0.0006417039154057605
    }
}
```


the error analysis corresponding to the initial data is recorded and the error of the first frame is disregarded when all the frames are considered in reproduce.