

COF-Landscaper Usage Guide

Workflow, parameters, examples, and outputs

Gregor Lauter

Contents

1	COF-Landscaper Usage Guide	2
1.1	What COF-Landscaper Does	2
1.2	Quick Start	2
1.2.1	Option A: Interactive notebook workflow	2
1.2.2	Option B: Hybrid script + JSON workflow	2
1.3	Required Inputs	3
1.3.1	Node and linker .xyz fragments	3
1.3.2	Dummy atoms and connection sites	3
1.3.3	Topology selection	3
1.3.4	Stacking mode selection	3
1.3.5	Optional experimental PXRD .xy file	3
1.4	Workflow Overview	3
1.5	Step-by-Step Workflow Reference	4
1.5.1	COF construction (<code>BuildCOF2D</code>)	4
1.5.2	Single-layer preoptimization (<code>MaceOpt.run_preopt</code>)	4
1.5.3	Stacking matrix generation (<code>CreateMatrix.run</code>)	6
1.5.4	MACE single-point energies (<code>MaceSP.run_mode</code>)	7
1.5.5	Optional DFT single-point input generation (<code>CrystalSP.generate_input</code>)	7
1.5.6	DFT single-point energy extraction (<code>CrystalSP.read_output</code>)	8
1.5.7	PES plotting (<code>Landscape.run_mode</code>)	8
1.5.8	Candidate selection (<code>SelectCofs.run_mode</code>)	9
1.5.9	Geometry optimization (<code>MaceOpt.run_mode</code>)	10
1.5.10	Optional DFT optimization input generation (<code>CrystalOpt.generate_input</code>)	10
1.5.11	Extract DFT optimized structures (<code>CrystalOpt.extract_cif</code> , <code>CrystalOpt.read_output</code>)	11
1.5.12	Analyze stacking (<code>AnalyzeStacking.analyze</code>)	11
1.5.13	PXRD simulation and plotting (<code>PXRD.run</code> , <code>PXRD.plot_sim</code> , <code>PXRD.plot_sim_vs_exp</code>)	12
1.5.14	Optional visualization (<code>VisualizeCOF.visualize_cof</code>)	13
1.6	Minimal Example	14
1.7	MACE heads and dispersion settings	14
1.8	Inputs and outputs	15
1.9	Troubleshooting	15
1.10	PDF export	16
1.11	Figure placeholders	16
1.12	Where to learn more	16

1 COF-Landscaper Usage Guide

This guide explains how to run the COF-Landscaper workflow, including the notebook workflow and the hybrid script + JSON workflow. Installation steps are in [README.md](#). This document focuses on how to use the software.

Note: This manual describes the current recommended workflow. The example notebook includes additional non-default settings and remains the most detailed reference for advanced usage.

1.1 What COF-Landscaper Does

COF-Landscaper is a Python workflow for building and screening stacked 2D COF structures. It:

- builds 2D COFs from node and linker fragments
- samples stacking configurations using ILD x ILS grids
- evaluates energies with MACE and optionally DFT
- plots potential energy landscapes (PES)
- selects candidate structures for refinement
- optimizes structures (MACE or DFT)
- analyzes relaxed ILD and ILS
- simulates PXRD patterns for comparison with experiment

1.2 Quick Start

Choose one of the two workflows below.

1.2.1 Option A: Interactive notebook workflow

Use this for manual inspection, explanations, plotting, and non-default workflows.

1. Open [examples/notebook/cof-landscaper.ipynb](#).
2. Set COF_NAME, TOPOLOGY, BOND_TYPE, MODE, and MACE_HEAD as needed.
3. Run the notebook cells for construction, preoptimization, matrix generation, PES screening, candidate selection, optimization, analysis, and PXRD.

1.2.2 Option B: Hybrid script + JSON workflow

Use this for batch execution or HPC-style runs.

1. Edit the parameter file [examples/hybrid/cof-landscaper.params.json](#).
2. Make sure your working directory contains 0_node/ and 0_linker/ with exactly one .xyz file each.
3. Run the workflow script:

```
python examples/hybrid/cof-landscaper.py
```

If you want to run from the example folder where the input fragments already exist:

```
cd examples/hybrid
python cof-landscaper.py
```

The notebook can be used afterward to load outputs and generate PXRD comparison plots without running additional compute stages.

1.3 Required Inputs

1.3.1 Node and linker .xyz fragments

- The workflow expects node and linker building units of the final COF layer, not necessarily the experimental precursors.
- No explicit reaction or fragmentation step is modeled.
- Fragments should be roughly relaxed before use (for example with UFF in a molecular editor).
- Default input locations are `0_node/` and `0_linker/` in the current working directory.

Warning: Provide roughly relaxed fragments to avoid severe steric clashes during stacking.

Warning: Highly distorted or strongly non-planar fragments can cause clashes when the stacking matrix is generated.

1.3.2 Dummy atoms and connection sites

Dummy atoms define connection sites in the fragments:

- **He** = single bond connection
- **Se** = double bond connection

Warning: The dummy atom type must match `BOND_TYPE`. If these are inconsistent, COF construction may fail.

1.3.3 Topology selection

Supported values for `TOPOLOGY`:

- **hcb**
- **sql**

1.3.4 Stacking mode selection

Supported values for `MODE`:

- **incl** (inclined stacking)
- **serr** (serrated stacking)
- **both** (generate both)

1.3.5 Optional experimental PXRD .xy file

Use experimental `.xy` data for `PXRD.plot_sim_vs_exp` comparisons. If `exp_xy_file` is not provided, the function searches for exactly one `.xy` file in an `experimental_pxrd` folder in the current working directory.

1.4 Workflow Overview

1. Build a single-layer COF
2. Preoptimize the layer with constrained z motion

3. Generate the ILD x ILS stacking matrix
4. Run MACE single-point energies
5. Plot the PES
6. Select candidate structures
7. Optimize candidates with MACE or DFT
8. Analyze relaxed stacking metrics
9. Simulate PXRD and compare to experiment

1.5 Step-by-Step Workflow Reference

1.5.1 COF construction (BuildCOF2D)

Builds one unoptimized single-layer COF from a node and a linker. When `bond_type` is provided (the normal workflow), it reads one `.xyz` file from each of `0_node/` and `0_linker/`.

Example:

```
builder = cl.BuildCOF2D()
builder.build(topo=TOPOLOGY, bond_type=BOND_TYPE, cof_name=COF_NAME)
```

Defaults:

Parameter	Default	Meaning
<code>topo</code>	required	Topology key (<code>hcb</code> or <code>sql</code>)
<code>bond_type</code>	required	Bond type (<code>single</code> or <code>double</code>)
<code>cof_name</code>	required	COF identifier for output naming
<code>input_node</code>	None	Uses <code>0_node/*.xyz</code> when <code>bond_type</code> is set
<code>input_linker</code>	None	Uses <code>0_linker/*.xyz</code> when <code>bond_type</code> is set
<code>output_folder</code>	None	Uses <code>{COF_NAME}/1_{COF_NAME}_single_layer</code>

Expected output:

```
{COF_NAME}/1_{COF_NAME}_single_layer/{COF_NAME}_unopt.cif
```

1.5.2 Single-layer preoptimization (MaceOpt.run_preopt)

Preoptimizes the single-layer COF while constraining out-of-plane motion. This keeps the layer approximately planar so the stacking matrix does not contain severe clashes.

Example:

```
preopt = cl.MaceOpt()
preopt.run_preopt(cof_name=COF_NAME)
```

Constructor defaults (`MaceOpt`):

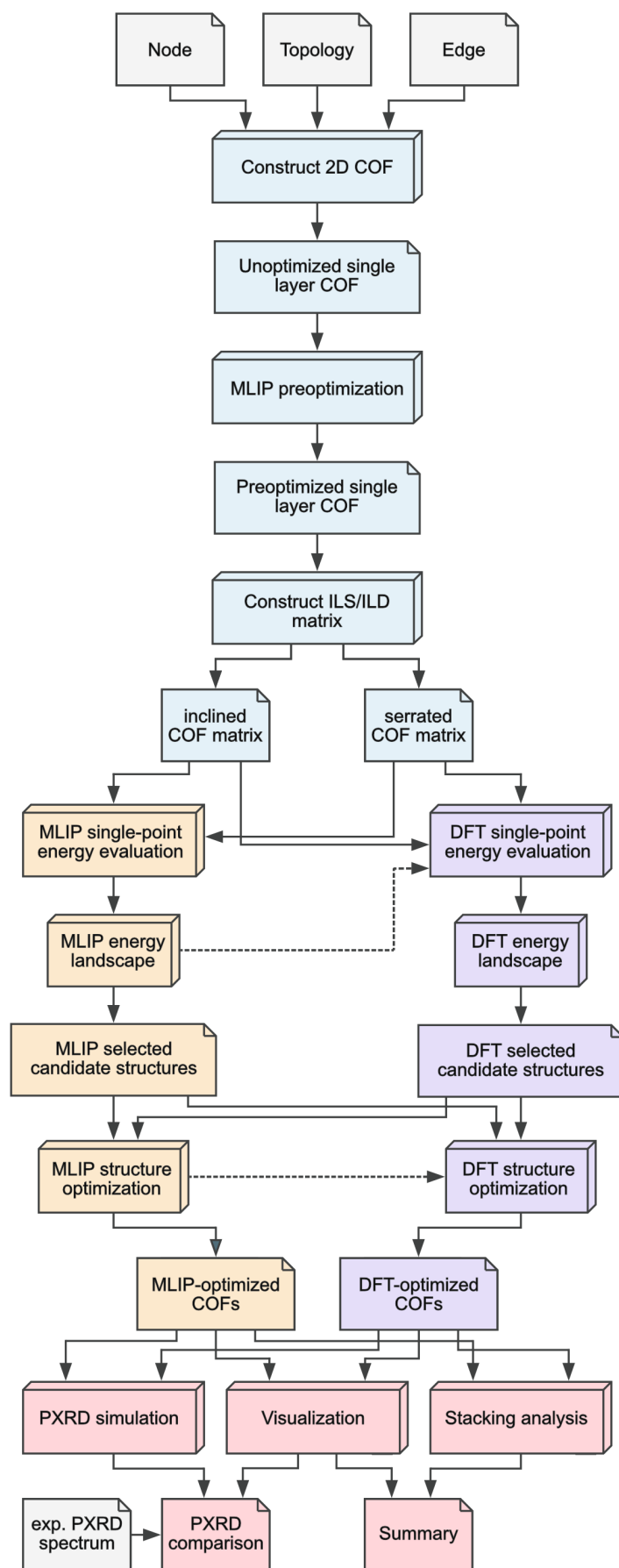


Figure 1: COF-Landscaper workflow overview

Parameter	Default	Meaning
<code>fmax</code>	0.01	Force convergence threshold in eV/Å
<code>dtype</code>	<code>float64</code>	Numerical precision
<code>head</code>	<code>spice_wB97M</code>	MACE head preset
<code>model</code>	<code>None</code>	MACE model ID (defaults to <code>mh-1</code>)
<code>device</code>	<code>cpu</code>	Compute device (<code>cpu</code> or <code>cuda</code>)
<code>fix_z</code>	<code>False</code>	Constrain z during general optimizations
<code>max_steps</code>	2000	Maximum optimizer steps
<code>verbose</code>	<code>True</code>	Write MACE init logs to <code>mace_calculator.log</code>

Method defaults (`run_preopt`):

Parameter	Default	Meaning
<code>input_path</code>	<code>None</code>	<code>{COF_NAME}/1_{COF_NAME}_single_layer/</code>
<code>output_path</code>	<code>None</code>	<code>{COF_NAME}/1_{COF_NAME}_single_layer/</code>
<code>fix_z</code>	<code>True</code>	Constrain z for preoptimization

Warning: Use `fix_z=True` only for preoptimization. For bulk optimization, set `fix_z=False`.

Expected output:

```
{COF_NAME}/1_{COF_NAME}_single_layer/{COF_NAME}_preopt.cif
```

1.5.3 Stacking matrix generation (`CreateMatrix.run`)

Generates an ILD x ILS grid by scaling the z lattice parameter (ILD) and applying in-plane slips (ILS). AA and AB are reference limits within the scan. Inclined and serrated modes are supported.

Example:

```
matrix = cl.CreateMatrix()
matrix.run(cof_name=COF_NAME, topo=TOPOLOGY, mode=MODE)
```

Defaults:

Parameter	Default	Meaning
<code>ild_start</code>	3.0	Minimum ILD (Ångström)
<code>ild_end</code>	4.0	Maximum ILD (Ångström)
<code>ild_step</code>	0.1	ILD step size (Ångström)
<code>ils_length_start</code>	0.0	Minimum ILS (AA reference)
<code>ils_length_end</code>	<code>None</code>	Auto-compute AB shift length
<code>ils_length_step</code>	1.0	ILS step size (Ångström)
<code>ils_angle</code>	<code>None</code>	Auto-compute AB shift angle
<code>print_shift</code>	<code>False</code>	Print default AB shift values
<code>input_cif</code>	<code>None</code>	<code>{COF_NAME}/1_{COF_NAME}_single_layer/</code>
<code>output_base_folder</code>	<code>None</code>	<code>{COF_NAME}/2_{COF_NAME}_matrix</code>

Parameter	Default	Meaning
-----------	---------	---------

Expected output:

```
{COF_NAME}/2_{COF_NAME}_matrix/{serr|incl}/
```

1.5.4 MACE single-point energies (`MaceSP.run_mode`)

Computes single-point energies for all stacked structures in the ILD x ILS grid. No relaxation occurs here, so the PES is a reduced-dimensional screening model.

Example:

```
sp = cl.MaceSP()
sp.run_mode(cof_name=COF_NAME, mode=MODE)
```

Constructor defaults (`MaceSP`):

Parameter	Default	Meaning
device	cpu	Compute device (cpu or cuda)
dtype	float64	Numerical precision
head	spice_wB97M	MACE head preset
model	None	MACE model ID (defaults to mh-1)
verbose	True	Write MACE init logs to mace_calculator.log

Method defaults (`run_mode`):

Parameter	Default	Meaning
input_folder	None	Uses {COF_NAME}/2_{COF_NAME}_matrix/{serr incl}
output_csv_dir	None	Uses {COF_NAME}/3_{COF_NAME}_landscape

Expected output:

```
{COF_NAME}/3_{COF_NAME}_landscape/{COF_NAME}_sp_energies_{serr|incl}.csv
```

1.5.5 Optional DFT single-point input generation (`CrystalSP.generate_input`)

Generates CRYSTAL23 .d12 input files for the stacked structures. These jobs must be run externally (HPC).

Example:

```
sp = cl.CrystalSP()
sp.generate_input(cof_name=COF_NAME, mode=MODE)
```

Defaults:

Parameter	Default	Meaning
basisset	SOLDEF2MSVP	CRYSTAL basis set
functional	HSESOL3C	Exchange-correlation functional
shrink	2 2 8	Monkhorst-Pack grid
post_block	None	Auto-generate BASISSET/DFT/SHRINK block
input_base_folder	None	{COF_NAME}/2_{COF_NAME}_matrix
output_base_folder	None	{COF_NAME}/2_{COF_NAME}_matrix

Warning: CRYSTAL23 jobs are not run by the workflow. Run them externally and place .out files next to their .d12 inputs before parsing.

Expected output:

```
{COF_NAME}/2_{COF_NAME}_matrix/dft_{serr|incl}/
```

1.5.6 DFT single-point energy extraction (CrystalSP.read_output)

Parses CRYSTAL23 .out files and writes _dft energy CSVs for PES analysis.

Example:

```
sp = cl.CrystalSP()
sp.read_output(cof_name=COF_NAME, mode=MODE)
```

Defaults:

Parameter	Default	Meaning
input_base_folder	None	{COF_NAME}/2_{COF_NAME}_matrix
output_base_folder	None	{COF_NAME}/3_{COF_NAME}_landscape

Warning: .out files must be placed in dft_{serr|incl} next to their .d12 inputs before parsing.

Expected output:

```
{COF_NAME}/3_{COF_NAME}_landscape/{COF_NAME}_sp_energies_{serr|incl}_dft.csv
```

1.5.7 PES plotting (Landscape.run_mode)

Generates heatmap and/or contour plots from the single-point CSVs and marks global or local minima.

Example:

```
landscape = cl.Landscape()
landscape.run_mode(cof_name=COF_NAME, mode=MODE)
```

Defaults:

Parameter	Default	Meaning
colorscheme	viridis	Matplotlib colormap
plot_mode	both	heatmap, isolines, or both
rel_energy_max	None	Clip relative energies above this value
show_minima_markers	True	Mark minima on the plot
minima_mode	global	Mark global or local minima
show_header	True	Draw title/header text
show_title_block	False	Add mode and level-of-theory lines
show	False	Interactive display
dft	False	Read _dft CSVs and label as DFT
input_folder	None	{COF_NAME}/3_{COF_NAME}_landscape
output_folder	None	{COF_NAME}/3_{COF_NAME}_landscape

Warning: The PES is a reduced-dimensional screening model. Always inspect structures and energies before committing to DFT refinement.

Expected output:

```
{COF_NAME}/3_{COF_NAME}_landscape/pes_{COF_NAME}_{serr|incl}_{heatmap|isolines}.png
```

1.5.8 Candidate selection (SelectCofs.run_mode)

Copies selected structures (global/local minima and optional manual picks) into optimization-ready folders. Use `selections_serr` and `selections_incl` to add explicit ILD/ILS points.

Example:

```
selector = cl.SelectCofs()
selector.run_mode(cof_name=COF_NAME, mode=MODE)
```

Defaults:

Parameter	Default	Meaning
selections_serr	None	Extra ILD/ILS points for serrated mode
selections_incl	None	Extra ILD/ILS points for inclined mode
include_autoselect	True	Include auto-selected minima
autoselect_minima	global	Select global or local minima
input_base	None	{COF_NAME}/2_{COF_NAME}_matrix
output_base	None	{COF_NAME}/3_{COF_NAME}_landscape/selections
input_folder	None	Optional single-mode override
output_folder	None	Optional single-mode override

Expected output:

```
{COF_NAME}/3_{COF_NAME}_landscape/selection/{serr|incl}/
```

1.5.9 Geometry optimization (MaceOpt.run_mode)

Optimizes candidate structures with full cell and atomic relaxation. Uses ASE `FrechetCellFilter` with LBFGS. Writes optimized CIFs and optional energy CSV.

Example:

```
opt = cl.MaceOpt()
opt.run_mode(cof_name=COF_NAME, mode=MODE)
```

Defaults:

Parameter	Default	Meaning
fmax	0.01	Force convergence threshold in eV/A
max_steps	2000	Maximum optimizer steps
dtype	float64	Numerical precision
head	spice_wB97M	MACE head preset
device	cpu	Compute device (cpu or cuda)
fix_z	False	Do not constrain z during bulk optimization
input_base	None	{COF_NAME}/3_{COF_NAME}_landscape/sele
output_base	None	{COF_NAME}/4_{COF_NAME}_optimization
save_opt_energies_csv	True	Write per-layer energy CSV

Expected output:

```
{COF_NAME}/4_{COF_NAME}_optimization/{serr|incl}/
{COF_NAME}/4_{COF_NAME}_optimization/{COF_NAME}_opt_energies_per_layer.csv
```

1.5.10 Optional DFT optimization input generation (CrystalOpt.generate_input)

Creates CRYSTAL23 .d12 inputs for geometry optimizations of selected structures.

Example:

```
opt = cl.CrystalOpt()
opt.generate_input(cof_name=COF_NAME, mode=MODE)
```

Defaults:

Parameter	Default	Meaning
basisset	SOLDEF2MSVP	CRYSTAL basis set
functional	HSESOL3C	Exchange-correlation functional
shrink	2 2 8	Monkhorst-Pack grid
maxtradius	0.5	OPTGEOM trust radius

Parameter	Default	Meaning
post_block	None	Auto-generate OPTGEOM/DFT blocks
input_base_folder	None	{COF_NAME}/3_{COF_NAME}_landscape/select
output_base_folder	None	{COF_NAME}/4_{COF_NAME}_optimization

Warning: CRYSTAL23 geometry optimizations are run externally. Place .out files next to their .d12 inputs before extraction.

Expected output:

```
{COF_NAME}/4_{COF_NAME}_optimization/dft_{serr|incl}/
```

1.5.11 Extract DFT optimized structures (CrystalOpt.extract_cif, CrystalOpt.read_output)

Parses CRYSTAL23 .out files to CIFs and writes DFT energy CSVs.

Example:

```
opt = cl.CrystalOpt()
opt.extract_cif(cof_name=COF_NAME, mode=MODE)
opt.read_output(cof_name=COF_NAME, mode=MODE)
```

Defaults (extract_cif):

Parameter	Default	Meaning
input_base_folder	None	{COF_NAME}/4_{COF_NAME}_optimization
output_base_folder	None	{COF_NAME}/4_{COF_NAME}_optimization

Defaults (read_output):

Parameter	Default	Meaning
input_base_folder	None	{COF_NAME}/4_{COF_NAME}_optimization
output_base_folder	None	{COF_NAME}/4_{COF_NAME}_optimization

Warning: CRYSTAL .out files must be placed next to their .d12 inputs before parsing.

Expected output:

```
{COF_NAME}/4_{COF_NAME}_optimization/dft_{serr|incl}/
{COF_NAME}/4_{COF_NAME}_optimization/{COF_NAME}_opt_energies_per_layer_dft.csv
```

1.5.12 Analyze stacking (AnalyzeStacking.analyze)

Computes relaxed ILD and ILS and writes summary CSVs. For serrated stacking, ILS uses a registry-based in-plane shift plus half of the tilt component. For inclined stacking, ILS is derived from the projection of the c vector onto the ab plane. ILD is halved for serrated bilayers.

Example:

```
analyzer = cl.AnalyzeStacking()
analyzer.analyze(cof_name=COF_NAME, mode=MODE)
```

Defaults:

Parameter	Default	Meaning
mode	both	Analyze serr, incl, or both
input_base	None	{COF_NAME}/4_{COF_NAME}_optimization
output_base	None	{COF_NAME}/5_{COF_NAME}_analysis
dft	False	Read dft_{mode} folders when True
print_values	True	Print ILD/ILS values to stdout

Expected output:

```
{COF_NAME}/5_{COF_NAME}_analysis/final_structures.csv
```

1.5.13 PXRD simulation and plotting (PXRD.run, PXRD.plot_sim, PXRD.plot_sim_vs_exp)

Simulates PXRD patterns from optimized CIFs and generates comparison plots.

Example (simulation):

```
pxrd = cl.PXRD(wavelength="CuKa", two_theta_range=(1.5, 60.0))
pxrd.run(cof_name=COF_NAME, mode=MODE)
```

Defaults (PXRD.run):

Parameter	Default	Meaning
wavelength	CuKa	X-ray wavelength preset
two_theta_range	(1.5, 60.0)	2theta range (deg)
mode	both	serr, incl, or both
dft	False	Use DFT-optimized structures
input_folder	None	{COF_NAME}/4_{COF_NAME}_optimization/ or dft_{mode}
output_folder	None	{COF_NAME}/5_{COF_NAME}_analysis/pxrd_ or pxrd_xy_dft

Defaults (plot_sim):

Parameter	Default	Meaning
xy_folder	None	Uses {COF_NAME}/5_{COF_NAME}_analysis/pxrd_ or pxrd_xy_dft
output_folder	None	Uses {COF_NAME}/5_{COF_NAME}_analysis
xlim	(1.5, 60.0)	2theta plot bounds

Parameter	Default	Meaning
<code>show</code>	<code>True</code>	Display plot
<code>save</code>	<code>True</code>	Save plot

Defaults (`plot_sim_vs_exp`):

Parameter	Default	Meaning
<code>exp_xy_file</code>	<code>None</code>	Searches <code>experimental_pxrd</code> for one <code>.xy</code> file
<code>simulated_xy_folder</code>	<code>None</code>	Uses <code>{COF_NAME}/5_{COF_NAME}_analysis/pxrd/</code> or <code>pxrd_xy_dft/{mode}</code>
<code>output_folder</code>	<code>None</code>	Uses <code>{COF_NAME}/5_{COF_NAME}_analysis</code>
<code>xlim</code>	<code>(1.5, 60.0)</code>	2theta plot bounds
<code>show</code>	<code>True</code>	Display plot
<code>save</code>	<code>True</code>	Save plot

Expected output:

```
{COF_NAME}/5_{COF_NAME}_analysis/pxrd_xy/{serr|incl}/
{COF_NAME}/5_{COF_NAME}_analysis/pxrd_xy_dft/{serr|incl}/
{COF_NAME}/5_{COF_NAME}_analysis/{COF_NAME}_sim_{serr|incl}.png
{COF_NAME}/5_{COF_NAME}_analysis/{COF_NAME}_{serr|incl|both}.png
```

1.5.14 Optional visualization (`VisualizeCOF.visualize_cof`)

Visualizes optimized structures in a notebook using `py3Dmol`.

Defaults:

Parameter	Default	Meaning
<code>mode</code>	<code>both</code>	<code>serr</code> , <code>incl</code> , or <code>both</code>
<code>input_base</code>	<code>None</code>	<code>{COF_NAME}/4_{COF_NAME}_optimization</code>
<code>dft</code>	<code>False</code>	Visualize DFT-optimized structures
<code>add_unit_cell</code>	<code>True</code>	Draw unit cell
<code>supercell_size_serr</code>	<code>(2, 2, 1)</code>	Supercell size for serrated mode
<code>supercell_size_incl</code>	<code>(2, 2, 2)</code>	Supercell size for inclined mode

Warning: `cuda` only works if a CUDA-enabled PyTorch/MACE installation is available on your system.

1.6 Minimal Example

This is based on the hybrid example in [examples/hybrid/cof-landscaper.params.json](#).

Example parameters:

Setting	Example value
COF_NAME	ILCOF-1
TOPOLOGY	sql
BOND_TYPE	single
MODE	both
MACE_HEAD	spice_wB97M
DEVICE	cuda
MAX_STEPS	3000
MINIMA_MODE	local

Node and linker fragments should be placed in:

```
examples/hybrid/0_node/  
examples/hybrid/0_linker/
```

Run the workflow:

```
cd examples/hybrid  
python cof-landscaper.py
```

Check that the first output exists:

```
ILCOF-1/1_ILCOF-1_single_layer/ILCOF-1_unopt.cif
```

1.7 MACE heads and dispersion settings

MACE calculations use the MACE-MH-1 model with the selected head preset.

MACE_HEAD	D3 correction	dispersion_xc	dispersion_cutoff	Notes
omat_pbe	yes	pbe	21.167088422553647	Trained without dispersion-corrected reference data
matpes_r2scan	yes	r2scan	40.0	Trained without dispersion-corrected reference data
omol	no	N/A	N/A	Trained on dispersion-inclusive reference data

MACE_HEAD	D3 correction	dispersion_xc	dispersion_cutoff	Notes
spice_wB97M	no	N/A	N/A	Trained on dispersion-inclusive reference data

1.8 Inputs and outputs

Stage	Output	Location pattern
Single-layer construction	Unoptimized CIF	{COF_NAME}/1_{COF_NAME}_single_layer/{COF_NAME}.cif
Preoptimization	Preoptimized CIF	{COF_NAME}/1_{COF_NAME}_single_layer/{COF_NAME}.cif
Matrix generation	Stacked CIFs	{COF_NAME}/2_{COF_NAME}_matrix/{serr incl}.cif
MACE energies	CSV	{COF_NAME}/3_{COF_NAME}_landscape/{COF_NAME}.csv
DFT energies	CSV	{COF_NAME}/3_{COF_NAME}_landscape/{COF_NAME}.csv
PES plots	PNGs	{COF_NAME}/3_{COF_NAME}_landscape/pes_{COF_NAME}.png (add _dft suffix for DFT)
Candidate selection	Selected CIFs	{COF_NAME}/3_{COF_NAME}_landscape/selected_{COF_NAME}.cif
MACE optimization	Optimized CIFs	{COF_NAME}/4_{COF_NAME}_optimization/{COF_NAME}.cif
MACE optimization	Energy CSV	{COF_NAME}/4_{COF_NAME}_optimization/{COF_NAME}.csv
DFT optimization inputs	CRYSTAL inputs	{COF_NAME}/4_{COF_NAME}_optimization/{COF_NAME}.inp
DFT optimization	Energy CSV	{COF_NAME}/4_{COF_NAME}_optimization/{COF_NAME}.csv
Final analysis	Summary CSV	{COF_NAME}/5_{COF_NAME}_analysis/final_{COF_NAME}.csv
PXRD	Simulated .xy	{COF_NAME}/5_{COF_NAME}_analysis/pxrd_{COF_NAME}.xy or pxrd_xy_dft/{serr incl}/pxrd_{COF_NAME}.xy
PXRD plots	PNGs	{COF_NAME}/5_{COF_NAME}_analysis/{COF_NAME}.png

1.9 Troubleshooting

Problem	Likely cause	Fix
COF construction fails	Dummy atoms do not match BOND_TYPE	Ensure He for <code>single</code> and Se for <code>double</code>
Matrix generation creates clashes	Layer is too distorted	Preoptimize and inspect input fragments
No GPU acceleration	CUDA not available or <code>device</code> set to <code>cpu</code>	Install CUDA-enabled PyTorch and set <code>device="cuda"</code>
DFT parsing fails	.out files not next to .d12 files	Move CRYSTAL outputs into the expected folder
PES plots missing	CSVs are not present in landscape folder	Run MACE or DFT single-point energies first
PXRD comparison missing	No experimental .xy file supplied	Provide <code>exp_xy_file</code> or add one file to <code>experimental_pxrd/</code>

1.10 PDF export

To generate a PDF version of this manual, run:

```
bash docs/readme/export_usage_pdf.sh
```

This writes `docs/readme/USAGE.pdf`. You can generate the PDF locally and commit it so users can download it directly.

1.11 Figure placeholders

Place future images in `docs/readme/figures/` and uncomment as needed.

1.12 Where to learn more

- Notebook workflow and explanations: [examples/notebook/cof-landscaper.ipynb](#)
- Hybrid workflow: [examples/hybrid/cof-landscaper.py](#) and [examples/hybrid/cof-landscaper.params.json](#)