*Supplementary Information*

# *Automated analysis of DFT output files for molecular descriptor extraction and reactivity modeling*

**Yu-Chien Huang,[a] Dennis Chung-Yang Huang,*[b,c] Yun-Cheng Tsai*[a]**

[a]*Department of Technology Application and Human Resource Development, National Taiwan Normal University, Taipei 106, Taiwan*
[b]*Department of Chemistry, Oklahoma State University, 107 Physical Sciences I, Stillwater, Oklahoma 74078, United States*
[c]*Institute for Chemical Reaction Design and Discovery (WPI-ICReDD), Hokkaido University, Kita 21, Nishi 10, Kita-ku, Sapporo, Hokkaido 001-0021, Japan*

*Email: dcyhuang@okstate.edu, pecu610@gmail.com

## Table of contents

## 1. General Information

This Supplementary Information consolidates the information that is currently split across multiple outputs into a single document. Please refer to the project repository for details: (https://github.com/peculab/DFTDescriptorPipeline)

## 2. Extracted Descriptor Set

All four case studies share the same core descriptor-extraction template at the single-aryl level. Pair-join cases (**Azoarene** and ***N,N′*-diarylindigo**) apply the same descriptor family to both aryl units, typically represented with ar1_ and ar2_ prefixes after merging. Single-join cases (**Heck–Boronic Acids** and ***N*-aryl-*N′*-alkylindigo**) use a single aryl descriptor block. This common framework provides a consistent basis for descriptor extraction, feature selection, and regression analysis across chemically distinct datasets.

**Energetic and frontier-orbital descriptors**

scf_energy_hartree, gibbs_free_energy_hartree, enthalpy_hartree, dipole_total_debye, isotropic_polarizability_au, homo_hartree, lumo_hartree, gap_hartree, ir_c_o_freq_cm1, ir_c_o_intensity_km_mol

**Atomic charge descriptors**

q_a, nbo_q_a, q_b, nbo_q_b, q_c, nbo_q_c, q_d, nbo_q_d, q_e, nbo_q_e, q_f, nbo_q_f, q_g, nbo_q_g, q_mean, q_std, nbo_q_mean, nbo_q_std

**Bond occupancy, bond energy, and bond-length descriptors**

nbo_bond_occ_c_a, nbo_bond_occ_c_d, nbo_bond_occ_c_e, nbo_bond_energy_c_a, nbo_bond_energy_c_d, nbo_bond_energy_c_e, bond_length_c_a, bond_length_c_d, bond_length_c_e

**Steric and aggregate aryl descriptors**

sterimol_L, sterimol_B1, sterimol_B5, Ar_NBO_C1, Ar_NBO_C2, Ar_NBO_OH, Ar_NBO_CO, Ar_v_CeqO, Ar_I_CeqO, Ar_Ster_L, Ar_Ster_B1, Ar_Ster_B5, Ar_dp, Ar_polar, Ar_HOMO, Ar_LUMO

## 3. Top-Performing Models by Case Study

The table below summarizes the best-performing models in each case study:

| Case Study | Method / Join | Target | Data Size | Best Subset | $R^2$ / $Q^2$ / RMSE |
|---|---|---|---|---|---|
| Azoarene | OLS / pair:Ar1+Ar2 | ln(kobs) | 25 / 25; k=4 | ar2_scf_energy_hartree; ar2_gibbs_free_energy_hartree; ar1_nbo_q_a; ar1_Ar_NBO_OH | 0.618 / 0.468 / 2.740 |
| Azoarene | SVR (RBF) / pair:Ar1+Ar2 | ln(kobs) | 25 / 25; k=5 | ar2_homo_hartree; ar1_bond_length_c_a; ar2_gap_hartree; ar2_bond_length_c_e; ar1_nbo_q_a | 0.815 / 0.486 / 1.907 |
| Heck–Boronic Acids | OLS / single:Ar | ΔΔG | 17 / 17; k=5 | bond_length_c_e; q_e; enthalpy_hartree; gibbs_free_energy_hartree; homo_hartree | 0.831 / 0.641 / 0.274 |
| *N*-aryl-*N*′-alkylindigo | OLS / single:Ar | ln(kobs)_MeCN | 21 / 21; k=5 | Ar_v_C=O; Ar_NBO_-O; LUMO; L_C1-C2; Ar_NBO_C2_x | 0.932 / 0.841 / 0.141 |
| *N*,*N*′-diarylindigo | OLS / pair:Ar1+Ar2 | ln(kobs) | 12 / 12; k=4 | ar1_nbo_q_b; ar2_bond_length_c_a; ar2_bond_length_c_d; Ar2_v_C=O | 0.989 / 0.976 / 0.191 |

Please refer to the following links for other top-performing models in each case study:

**Azoarene**

https://github.com/peculab/DFTDescriptorPipeline/blob/main/results/azoarene_experiments/ols_baseline/azoarene_top_models_by_k.csv

**Heck–Boronic Acids**

https://github.com/peculab/DFTDescriptorPipeline/blob/main/results/heck_boronic_acids/heck_boronic_acids_top_models_by_k.csv

***N*-aryl-*N*′-alkylindigo**

https://github.com/peculab/DFTDescriptorPipeline/blob/main/results/indigo_aryl_alkyl/indigo_aryl_alkyl_rerun_clean_top_models_by_k.csv

***N*,*N*′-diarylindigo**

https://github.com/peculab/DFTDescriptorPipeline/blob/main/results/indigo_diaryl/indigo_diaryl_top_models_by_k.csv

## 4. Runtime Summary

The table below summarizes the runtime in each case study:

| Case Study | Best model reported | Runtime (s) |
|---|---|---|
| Azoarene | OLS best subset | 404.4517 |
| Heck–Boronic Acids | OLS best subset | 310.8977 |
| *N*-aryl-*N*′-alkylindigo | OLS best subset | 363.4816 |
| *N*,*N*′-diarylindigo | OLS best subset | 224.7821 |

**Hardware specifications**

CPU: Intel Core i7-12700H

RAM: 16 GB

Storage: NVMe SSD

GPU 1: Intel Iris Xe Graphics

GPU 2: NVIDIA GeForce RTX 3050 Laptop GPU

## 5. Descriptor Correlations and Feature-Importance-style Diagnostics

The tables below summarize in each case study (i) the Pearson correlation of each selected descriptor with the target and (ii) the absolute standardized regression coefficient as a simple feature-importance-style measure:

**Azoarene (OLS best subset)**

| Descriptor | Pearson r with target | Abs. standardized coefficient | Interpretation |
|---|---|---|---|
| ar2_scf_energy_hartree | -0.580 | 6832.121 | energy-related |
| ar2_gibbs_free_energy_hartree | -0.580 | 6830.251 | energy-related |
| ar1_Ar_NBO_OH | -0.516 | 1.105 | aryl NBO/OH electronic |
| ar1_nbo_q_a | -0.516 | 1.105 | aryl NBO electronic |

**Heck–Boronic Acids (OLS best subset)**

| Descriptor | Pearson r with target | Abs. standardized coefficient | Interpretation |
|---|---|---|---|
| bond_length_c_e | 0.818 | 0.286 | bond length |
| q_e | -0.467 | 0.315 | atomic charge |
| enthalpy_hartree | -0.467 | 7625.102 | enthalpy |
| gibbs_free_energy_hartree | -0.467 | 7624.967 | free energy |
| homo_hartree | -0.444 | 0.214 | frontier orbital |

***N*-aryl-*N'*-alkylindigo (OLS best subset)**

| Descriptor | Pearson r with target | Abs. standardized coefficient | Interpretation |
|---|---|---|---|
| Ar_v_C=O | -0.721 | 0.491 | carbonyl vibration |
| Ar_NBO_-O | -0.654 | 0.357 | oxygen-site NBO |
| LUMO | 0.622 | 0.173 | frontier orbital |
| L_C1-C2 | -0.323 | 0.818 | C1–C2 bond length |
| Ar_NBO_C2_x | -0.155 | 0.467 | aryl C2 NBO x |

***N,N′*-diarylindigo (OLS best subset)**

| Descriptor | Pearson r with target | Abs. standardized coefficient | Interpretation |
|---|---|---|---|
| ar1_nbo_q_b | -0.944 | 1.124 | Ar1 NBO charge |
| ar2_bond_length_c_a | 0.871 | 0.598 | Ar2 bond length |
| ar2_bond_length_c_d | 0.870 | 2.643 | Ar2 bond length |
| Ar2_v_C=O | -0.862 | 2.369 | Ar2 carbonyl vibration |

## 6. Step-by-Step User Guide & Troubleshooting Instructions

### 1. Purpose of this guide

This guide explains how to run the **DFTDescriptorPipeline** from start to finish. It is written for users who want a clear, reproducible workflow, including:

- preparing the folder structure,
- installing dependencies,
- running the full pipeline,
- understanding the output files,
- adjusting the regression settings,
- and troubleshooting common problems.

The current pipeline is designed for **Gaussian log files** and supports the four case-study layouts used in this project.

### 2. What the pipeline does

The pipeline performs the following steps:

1. **Reads Gaussian .log files**
2. **Finds atom labels** around the target motif and generates a mapping table
3. **Extracts descriptors** from each log file
4. **Loads the experimental data** from Excel
5. **Matches descriptors with experimental entries**
6. **Builds regression models** by subset search
7. **Generates plots, reports, and summary files**

In practical terms, this means the pipeline turns raw Gaussian outputs into:

- descriptor tables,
- regression results,
- model summaries,
- feature analysis files,
- and timing records.

### 3. Expected project structure

Place the scripts and datasets under one project folder. A typical layout is:

DFTDescriptorPipeline/

batch_runner.py

```
extract_abcefg_from_logs.py
extractor_regr.py
requirements.txt
README.md

azoarene/
  logfiles/
    *.log
  azoarene_data.xlsx

heck_boronic_acids/
  logfiles/
    *.log
  heck_boronic_acids_data.xlsx

indigo_aryl_alkyl/
  logfiles/
    *.log
  indigo_aryl_alkyl_data.xlsx

indigo_diaryl/
  logfiles/
    *.log
  indigo_diaryl_data.xlsx
```

**Important notes**

- Each case-study folder must contain a **logfiles/** subfolder.
- Each case-study folder must also contain the matching **Excel file**.
- The names and paths must match the configuration in batch_runner.py.

**4. Software requirements**

You need:

- Python 3.10 or newer recommended

- pip
- the packages listed in requirements.txt

The pipeline relies on common Python libraries such as:

- pandas
- numpy
- plotly
- scikit-learn

**5. Installation**

**Windows (PowerShell)**

python -m venv .venv

.\.venv\Scripts\Activate.ps1

pip install -r requirements.txt

**macOS / Linux**

python3 -m venv .venv

source .venv/bin/activate

pip install -r requirements.txt

After installation, make sure all required packages were installed successfully before running the pipeline.

**6. Step-by-step workflow**

**Step 1. Prepare the Gaussian log files**

Copy all Gaussian output files into the corresponding logfiles/ folder for each case study.

Example:

azoarene/logfiles/

heck_boronic_acids/logfiles/

indigo_aryl_alkyl/logfiles/

indigo_diaryl/logfiles/

**Recommendations**

- Keep the log filenames consistent and clean.
- Avoid duplicate names that refer to different molecules.
- Make sure the log files contain the sections needed for descriptor extraction, such as:
    - SCF energy

- HOMO/LUMO information
- Mulliken charges
- thermodynamic values if those are required in your workflow

**Step 2. Prepare the experimental Excel files**

Each case-study folder must contain its Excel data file. The target column is defined in batch_runner.py.

Current configuration:

- azoarene → target: ln(kobs)
- heck_boronic_acids → target: ddG
- indigo_aryl_alkyl → target: ln(kobs)_MeCN
- indigo_diaryl → target: ln(kobs)

**Recommendations**

- Make sure the target column exists exactly as written.
- Keep the molecule identifiers consistent with the naming used in the log files.
- For pair-style datasets, confirm that the necessary identifier columns are present.

**Step 3. Check the case-study configuration**

Open batch_runner.py and verify:

- the case-study names,
- the log_subdir values,
- the Excel file paths,
- and the target column names.

You only need to change these settings if:

- your folder names are different,
- your Excel files have different names,
- or you want to add a new case study.

**Step 4. Run the full pipeline**

From the project root, run:

python batch_runner.py --base .

If you are running from another directory, provide the absolute path:

python batch_runner.py --base "C:\path\to\DFTDescriptorPipeline"

This command will:

1. run the atom-mapping step,
2. run the regression step,
3. generate case-level outputs,
4. and write summary files under results/.

**Step 5. Optional run modes**

**Skip the mapping step**

Use this if the mapping CSV files were already generated and you only want to rerun the modeling part.

python batch_runner.py --base . --skip-mapping

**Skip the regression step**

Use this if you only want to regenerate mapping outputs.

python batch_runner.py --base . --skip-regression

**Auto-open HTML plots**

python batch_runner.py --base . --open-plots

**7. Main output files**

The pipeline writes both **case-specific files** and **summary files**.

**7.1 Mapping output**

For each case, the mapping step produces:

- {case}_mapping.csv

This file typically includes:

- logfile
- atom labels such as a, b, c, d, e, f, g
- success flag
- error message if mapping failed
- elapsed time

This file is useful for checking whether atom labeling worked correctly.

**7.2 Modeling outputs**

For each case and each feature subset size, the pipeline may generate files such as:

- {case}_k{k}_features.csv

- {case}_k{k}_subset_search.csv
- {case}_k{k}_top_models_by_k.csv
- {case}_k{k}_Regression_Plot.html
- {case}_k{k}_Coef_Plot.html
- {case}_k{k}_regression_report.json

**What these files mean**

**features.csv**

- the merged modeling table
- descriptors and target values used for regression

**subset_search.csv**

- the models that passed the defined thresholds
- useful for comparing candidate models

**top_models_by_k.csv**

- the best-ranked models for a given feature size

**Regression_Plot.html**

- interactive plot of actual vs predicted values
- includes model metrics and the fitted equation

**Coef_Plot.html**

- coefficient plot for linear models

**regression_report.json**

- machine-readable summary of the selected model and run settings

### 7.3 Additional analysis outputs

Depending on the current script version and run settings, the workflow may also generate:

- descriptor-correlation tables
- feature-importance tables
- timing summaries
- case-level summary files
- an all-case summary table under results/

These files are especially useful for:

- supplementary information,
- and model interpretability analysis.

## 8. Understanding the regression settings

The main regression behavior is controlled by **environment variables**.

### 8.1 Cross-validation settings

**LOOCV**

CV_MODE=loocv

- more exhaustive
- usually slower
- useful for small datasets

**K-fold CV**

CV_MODE=kfold

N_SPLITS=5

- usually faster than LOOCV
- useful when you want a more practical runtime

### 8.2 Search size and speed settings

These variables control how many feature combinations are tested.

CAND_POOL=25

MAX_COMBOS=250000

SAMPLE_COMBOS=80000

PREFILTER_TOP=0

**Meaning**

- CAND_POOL: size of the candidate descriptor pool
- MAX_COMBOS: maximum number of combinations to enumerate fully
- SAMPLE_COMBOS: number of combinations to sample if the search space becomes too large
- PREFILTER_TOP: optional quick prefilter before expensive CV evaluation

**Practical advice**

- Use a smaller CAND_POOL for faster test runs.
- Use a larger CAND_POOL for more thorough exploration.
- If runtime is too long, reduce the search space first.

### 8.3 Model settings

The currently documented options in the available script are:

REGRESSOR=ols

or

REGRESSOR=ridge

RIDGE_ALPHA=1.0

**Meaning**

- ols: ordinary least squares
- ridge: ridge regression with regularization

If you maintain a custom local branch that adds other regressors, document those options in the same way before sharing the workflow with other users.

### 8.4 Model quality thresholds

MIN_R2=0.75

MIN_Q2=0.75

These thresholds decide which models are considered acceptable in the subset search output.

**Practical advice**

- Lower the thresholds if no model passes.
- Keep them strict if you want to report only strong models.

## 9. Example commands

**Example 1. Default full run**

python batch_runner.py --base .

**Example 2. Faster run with k-fold CV**

**Windows PowerShell**

$env:CV_MODE="kfold"

$env:N_SPLITS="5"

$env:CAND_POOL="20"

python batch_runner.py --base .

**macOS / Linux**

export CV_MODE=kfold

export N_SPLITS=5

export CAND_POOL=20

python batch_runner.py --base .

**Example 3. Ridge regression run**

**Windows PowerShell**

```
$env:REGRESSOR="ridge"
$env:RIDGE_ALPHA="10.0"
python batch_runner.py --base .
```

**macOS / Linux**

```
export REGRESSOR=ridge
export RIDGE_ALPHA=10.0
python batch_runner.py --base .
```

## 10. How to inspect the results

After the run finishes, check the outputs in this order.

### 10.1 First, inspect the mapping CSV

Ask:

- Were the atom labels assigned successfully?
- Are there failed rows?
- Do the failed rows show a clear error message?

If the mapping step fails for many files, fix that before trusting the regression results.

### 10.2 Second, inspect the merged feature table

Open:

- {case}_k{k}_features.csv

Ask:

- Did the descriptors merge correctly?
- Are the target values present?
- Are there too many missing values?

### 10.3 Third, inspect the top model table

Open:

- {case}_k{k}_top_models_by_k.csv

Ask:

- Which descriptor subsets perform best?
- Are the top models stable?
- Do the selected descriptors make chemical sense?

### 10.4 Fourth, inspect the interactive plot

Open:

- {case}_k{k}_Regression_Plot.html

Ask:

- Does the actual-vs-predicted trend look reasonable?
- Are there obvious outliers?
- Are the reported metrics acceptable?

**10.5 Finally, inspect JSON and summary files**

Use these for:

- manuscript reporting,
- supplementary tables,
- reproducibility records.

## 11. Typical workflow for a new dataset

If you want to test a new reaction system, follow this sequence.

1. Create a new dataset folder.
2. Add a logfiles/ subfolder.
3. Put the Gaussian log files inside.
4. Add the Excel file with the experimental targets.
5. Register the new case in batch_runner.py.
6. Set the target column.
7. Run the full pipeline.
8. Check the mapping file.
9. Check the merged descriptor table.
10. Review the top models and plots.
11. Record the final settings used in the run.

This is the easiest way to keep the workflow reproducible.

## 12. Troubleshooting

**Problem 1. The pipeline cannot find the log folder**

**Possible reason**

The path in batch_runner.py does not match your actual folder structure.

**Fix**

Check:

- --base
- log_subdir
- case-study folder names

**Problem 2. The pipeline cannot find the Excel file**

**Possible reason**

The file is missing or the filename does not match the configuration.

**Fix**

Verify the exact filename and path in batch_runner.py.

**Problem 3. A required target column is missing**

**Possible reason**

The target column name in the Excel file does not match the configured name.

**Fix**

Rename the column in the Excel file or update the case configuration.

**Problem 4. Mapping failed for many logs**

**Possible reason**

The target motif was not detected consistently, or the log file format differs from what the parser expects.

**Fix**

- inspect a few failed log files manually,
- compare them with successful examples,
- and confirm that the relevant Gaussian sections exist.

**Problem 5. No model passes the thresholds**

**Possible reason**

The thresholds are too strict, the dataset is too noisy, or the candidate pool is not appropriate.

**Fix**

Try one or more of the following:

- lower MIN_R2 and MIN_Q2,
- reduce the feature subset size,

- adjust CAND_POOL,
- inspect outliers,
- and verify that the descriptor-target merge is correct.

**Problem 6. Runtime is too long**

**Possible reason**

The subset search space is too large or the CV mode is expensive.

**Fix**

Try:

- CV_MODE=kfold
- smaller CAND_POOL
- smaller SAMPLE_COMBOS
- nonzero PREFILTER_TOP

## 13. Reproducibility checklist

Before reporting final results, make sure you record:

- Python version
- package versions
- case-study configuration
- environment variables used
- target column name
- cross-validation mode
- candidate-pool size
- final model settings
- output folder location

## 14. Recommended wording for manuscript support files

If you want to describe this workflow briefly in a supplementary document, you may use the following wording:

The DFTDescriptorPipeline was executed from the project root using batch_runner.py. For each case study, Gaussian log files were parsed to extract electronic descriptors, atom-level mapping tables were generated, descriptor tables were merged with experimental data from Excel, and subset-search regression models were evaluated under user-defined cross-validation and threshold settings. The workflow generated case-level regression reports, interactive plots, and summary files for downstream analysis.