README Documentation

Fast Predictions of Liquid-Phase Acid-Catalyzed Reaction Rates Using Molecular Dynamics Simulations and Convolutional Neural Networks

Alex K. Chew[a], Shengli Jiang[a], Weiqi Zhang[a], Victor Zavala[a], and Reid C. Van Lehn[a†]

[†]Please send all e-mail correspondence to: vanlehn@wisc.edu
[a]Department of Chemical and Biological Engineering, University of Wisconsin-Madison, Madison, USA.

January 24, 2021

Contents

1 Description

The purpose of this document is to go through a step-by-step procedure in developing data for the article:

A. K. Chew, S. Jiang, W. Zhang, V. M. Zavala, and R. C. Van Lehn. "Fast predictions of liquid-phase acid-catalyzed reaction rates using molecular dynamics and convolutional neural networks." *Chemical Science*, **2020**, *11*, 12464-12476. [Link]

1.1 Main idea

In this work, we combined classical molecular dynamics (MD) simulations and 3D convolutional neural networks (CNN) to make predictions of experimental kinetic solvent parameters (σ). Figure 1 shows the general workflow that was outline in the paper. First, classical MD simulations were performed to generate solvent configurations around a reactant. We then converted MD trajectories into voxel representations by computing the occupancy of water, oxygens of reactant, and cosolvent atomic positions within $(0.2\ \text{nm})^3$ volume elements, stored as red, green, and blue channels, respectively. Each MD configuration resulted in a $20 \times 20 \times 20 \times 3$ array. We normalized each array by dividing the maximum number of atoms per channel. We averaged 2 ns of simulation data (corresponding to 200 MD configurations) to output a voxel representation, shown in Figure 1a. We then augmented the data, as discussed in the main text. The voxel representations were used as inputs to SolventNet, a 3D CNN developed in-house shown in Figure 1b. By combining classical MD simulations and 3D CNNs, we utilize highly computationally efficient methods to make accurate predictions σ, opening avenues to fast screening of solvent compositions and integration of classical MD with process model tools.

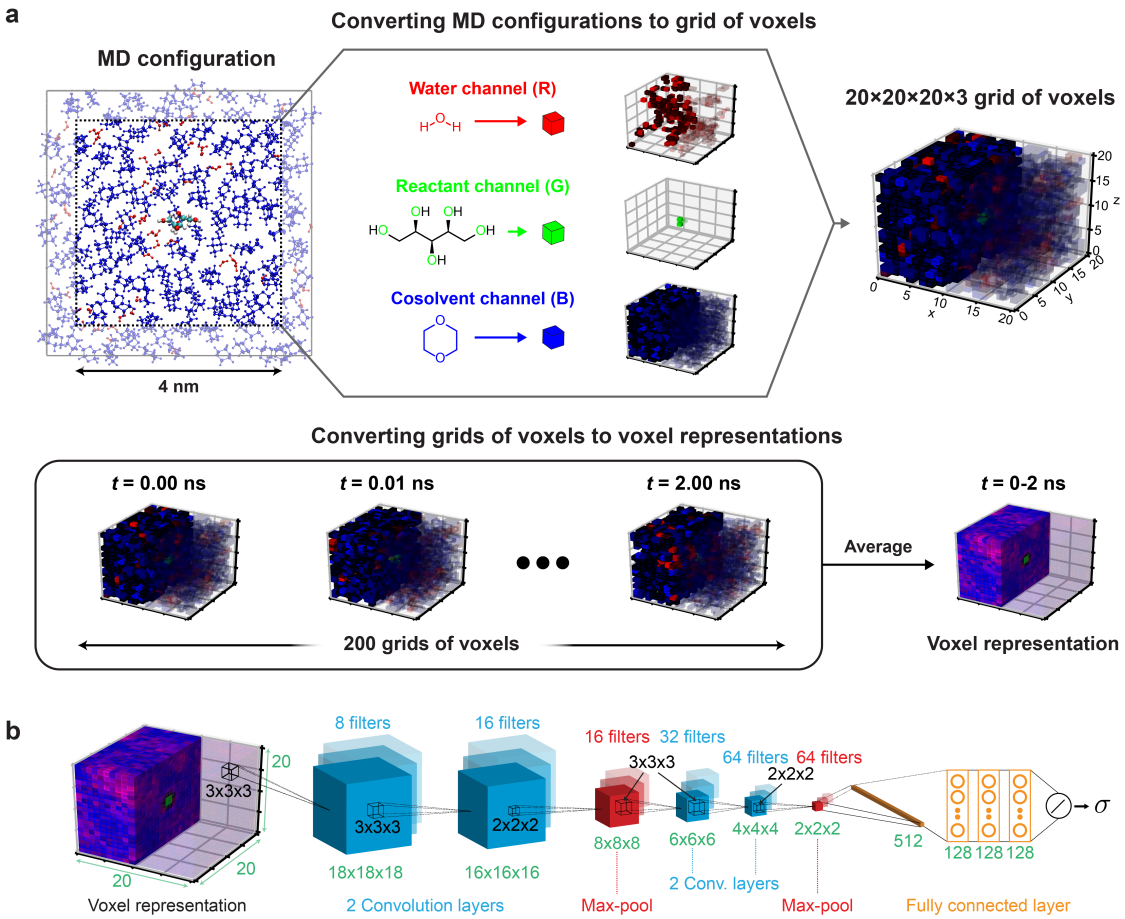

Figure 1: General workflow for using classical molecular dynamics simulations and 3D convolutional neural networks to make predictions of experimental kinetic solvent parameters (σ). (**a**) Conversion of MD configurations into voxel representations. (**b**) SolventNet architecture that interprets voxel representations and outputs σ. Additional details can be found in the main text.

1.2 Data availability

All classical MD simulations, 3D CNN architectures, and scripts are available here. To access the data, download the file, and unzip the directory. The command to unzip in Linux terminal is:

```
tar -zxvf NAME.tar.gz
```

where "NAME" is the name of the zipped file.

1.3 Software requirements

The following software is required to run the code:

- GROMACS 2016 (Version 0) - (Optional) Used to run MD simulations and convert MD trajectories.
- Python $\geq$ 3.6.8 - used to train 3D CNNs and make predictions with the modules and their versions listed:
  - `tensorflow` version 1.15.0
  - `keras` version 2.3.1
  - `scipy` version 0.18.1
  - `pandas` version 0.19.2
  - `matplotlib` version 2.0.0

  In this work, we used Python Version 3.6.8 to train and analyze the results.

1.4 Installing Python environment

For accessing the python environment, install Anaconda in the [Link]. Installing this should enable `conda` in the command line. To install the Python environment, run the Listing 1.

```
# CREATE PYTHON ENVIRONMENT
conda create -n py36_tensorflow2 python=3.6.8
# ACTIVATE ENVIRONMENT
conda activate py36_tensorflow2
# INSTALLING TENSORFLOW
pip install tensorflow==1.15.0
pip install keras==2.3.1
# INSTALL MODULES
pip install numpy sklearn matplotlib pandas numpy
# USE CONDA TO INSTALL MDTRAJ
conda install -n py36_tensorflow2 cython
conda install -n py36_tensorflow2 mdtraj
# INSTALLING IDE: SPYDER
conda install -n py36_tensorflow2 spyder==3.3.1
# TO ACTIVATE SPYDER:
spyder --new-instance
```

Listing 1: Installation of Python environment

1.5 Directory structure

The directories are within the main directory are listed and described below:

- `cnn_output`: output folders for 3D CNN training.
- `csv_output`: output folder for csv files.
- `combined_data_set`: pickle files containing combined training datasets.
- `database`: pickle files containing voxel representations that are extracted from MD trajectories.
- `python_scripts`: python scripts that were used for training 3D CNNs.

- `scripts`: bash scripts that use the python scripts to either convert MD simulations to voxel representations or train 3D CNNs.
- `storage`: location to store pickle files when cross validating or predicting kinetic solvent parameters.
- `MD_Simulations`: folder that contains all MD simulations.
- `test_set_data`: location to store test set data pickles.

2 Classical molecular dynamics simulations

This section describes the classical MD simulations used in the main text. All simulation data is available in the following directory:

`MAIN_dir/MD_simulations`

2.1 Directory structure

The `MAIN_dir/MD_simulations` directory contains the following files:

- `training_set`: Simulations containing 7 biomass-relevant model reactants: ethyl *tert*-butyl ether (ETBE), *tert*-butanol (TBA), 1,2-propanediol (PDO), levoglucosan (LGA), fructose (FRU), cellobiose (CEL), and xylitol (XYL). Solvent systems include aqueous mixtures with 25 wt%, 50 wt%, 75 wt%, or 90 wt% of one of three polar aprotic cosolvents: 1,4-dioxane (DIO), γ-valerolactone (GVL), and tetrahydrofuran (THF). These simulation contains the first 20 ns of production trajectories and were used to train 3D CNNs.
- `testing_set`: Simulations containing TBA, PDO, and FRU in DMSO-water mixtures and acetonitrile (MeCN)-water mixtures. In addition, this folder contains simulations of FRU and glucose (GLU) in aqueous mixtures with acetone (ACE). These simulations contain 4 ns of production trajectories and were used to test the prediction accuracy of fully trained 3D CNNs.

2.2 MD data used for training 3D CNNs

MD productions simulations containing the first 20 ns trajectories taken from Ref. [1] are available at:

`MAIN_dir/MD_simulations/training_set`

Full 200 ns production trajectories are available in this link. Listing 2 shows a list of the directories within `training_set`.

```
CEL
ETBE
FRU
LGA
PDO
tBuOH
XYL
```

Listing 2: Directories within `MD_simulations/training_set`

Each directory is labeled by the reactant name:

- `CEL`: cellobiose
- `ETBE`: ethyl *tert*-butyl ether
- `FRU`: fructose
- `LGA`: levoglucosan
- `PDO`: 1,2-propanediol
- `tBuOH`: *tert*-butanol
- `XYL`: xylitol

Each directory contains simulations for the specified reactant. An example for xylitol is shown in Listing 3.

```
mdRun_403.15_6_nm_XYL_10_WtPercWater_spce_dioxane
mdRun_403.15_6_nm_XYL_10_WtPercWater_spce_GVL_L
mdRun_403.15_6_nm_XYL_10_WtPercWater_spce_tetrahydrofuran
mdRun_403.15_6_nm_XYL_25_WtPercWater_spce_dioxane
mdRun_403.15_6_nm_XYL_25_WtPercWater_spce_GVL_L
mdRun_403.15_6_nm_XYL_25_WtPercWater_spce_tetrahydrofuran
mdRun_403.15_6_nm_XYL_50_WtPercWater_spce_dioxane
mdRun_403.15_6_nm_XYL_50_WtPercWater_spce_GVL_L
mdRun_403.15_6_nm_XYL_50_WtPercWater_spce_tetrahydrofuran
mdRun_403.15_6_nm_XYL_75_WtPercWater_spce_dioxane
mdRun_403.15_6_nm_XYL_75_WtPercWater_spce_GVL_L
mdRun_403.15_6_nm_XYL_75_WtPercWater_spce_tetrahydrofuran
```

Listing 3: Example of directories listed in `XYL`

The directory name was designed to inform about the simulation parameters. For example, `mdRun_403.15_6_nm_XYL_10_WtPercWater_spce_dioxane` means XYL was simulated with 10 wt% water/90 wt% dioxane at 403.15 K with the initial box length of 6 nm. Each directory contains the following main files:

- `mixed_solv_prod.gro`: Structure file
- `mixed_solv_prod_first_20_ns_centered_with_10ns.xtc`: *NPT* production simulation containing 20 ns worth of production data.
- `mixed_solv.top`: topology information
- `*.itp`: molecular information

The `mixed_solv_prod.gro` and `mixed_solv_prod_first_20_ns_centered_with_10ns.xtc` was used to generate voxel representations for training 3D CNNs, described in Section 3.1.

2.3 MD data in DMSO-, MeCN, and ACE mixtures

We use the same protocol as discussed in Ref. [1] to generate classical MD simulations of reactants in DMSO-, MeCN-, and ACE-water mixtures. These simulations are available in the following directory:

`MAIN_dir/MD_simulations/testing_set`

This directory contains the following directories:

- `DMSO`: MD simulations of *tert*-butanol as `tBuOH`, 1,2-propanediol as `PDO`, and fructose as `FRU` in DMSO-water mixtures.
- `MeCN`: MD simulations of *tert*-butanol as `tBuOH`, 1,2-propanediol as `PDO`, and fructose as `FRU` in MeCN-water mixtures.
- `ACE`: MD simulations of fructose as `FRU` and glucose as `GLU` in ACE-water mixtures.

Similar to Section 2.2, the names of within these directories were designed to inform about the simulation parameters. For example, in DMSO, `mdRun_363.15_6_nm_tBuOH_10_WtPercWater_spce_dmso` is a MD simulation of TBA in 10 wt% water/90 wt% DMSO, where 363.15 K is the temperature of the simulation and 6 nm is used as the initial simulation box length. These directories contain all simulation parameters necessary to run in GROMACS. The description of files is: (All asterisks indicates expansion found in unix. For example, `*_em.*` means all files containing "_em".)

- `charmm36-nov2016.ff`: force field file for CHARMM36
- `mixed_solv_prod.gro`: structure file for the production trajectory
- `mixed_solv_prod_first_4_ns_centered.xtc`: 4 ns *NPT* production trajectory with reactant centered
- Note, energy minimization files (`*_em.*`) and 500 ps *NPT* equilibration files (`*_equil.*`) were omitted from the dataset, but you could recreate them using the `submit.sh` script.

The `mixed_solv_prod_first_4_ns_centered.xtc` is the trajectory used to generate voxel representations for predictions using 3D CNNs.

3 Using 3D CNNs to analyze MD trajectories

This section describes how MD trajectories are converted into voxel representations, training 3D CNNs, and making predictions with fully trained 3D CNNs.

3.1 Converting MD trajectories into voxel representations

MD trajectories were converted to voxel representations using the following procedure:

- Open the following script:

  `MAIN_dir/scripts/loop_grid_interpolation.sh`

- Adjust the following variables shown in Listing 4.

```
## FOR DEVELOPING TRAINING SET
declare -a data_list=("All")
xtc_file="mixed_solv_prod_first_20_ns_centered_with_10ns.xtc" # Using first 20
    ns of data
path_to_sim="${parent_dir}/MD_Simulations/training_set"

# Uncomment below for testing set
## FOR DEVELOPING TEST SET
# declare -a data_list=("DMSO" "ACE" "ACN")
# xtc_file="mixed_solv_prod_first_4_ns_centered.xtc"

## DEFINING PATHS
path_to_training="${parent_dir}/MD_Simulations/training_set"
path_to_testing="${parent_dir}/MD_Simulations/testing_set"

############################
### LOOP GRIDING DETAILS ###
############################

# BOX INFORMATION
box_size=4.0
box_inc=0.2 # 20 x 20 x 20
# box_inc=0.25 # 16 x 16 x 16
# box_inc=0.125 # 32 x 32 x 32

## DEFINING MAPPING TYPE
box_map_type="3channel_oxy"
```

Listing 4: Adjustable variables in `loop_grid_interpolation.sh`

`data_list` refers to the MD data that is being converted: "All" means the training data described in Section 2.2; "DMSO", "ACE", and "MeCN" means the DMSO, ACE, and MeCN data described in Section 2.3. `box_size` is the box length around the reactant that is converted into voxel representations, which was 4 nm box length in the main text. `box_inc` is the length of a volume element, which was 0.2 nm in the main text. `box_map_type` is the type of voxel representation, which we varied to see the influence of input representations on the prediction accuracy of SolventNet. Below list the different types of voxel representations:

- `solvent_only`: 2 channel voxel representation consisting of water and cosolvent atoms in each bin
- `3channel_oxy`: 3 channel voxel representation consisting of water, oxygens of reactant, and cosolvent atoms in each bin. This is the representation used in the main text.
- `3channel_hydroxyl`: 3 channel voxel representation consisting of water, hydroxyls of reactant, and cosolvent atoms in each bin
- `allatom`: 3 channel voxel representation consisting of water, reactant, and cosolvent atoms in each bin
- `allatomwithsoluteoxygen`: 4 channel voxel representation consisting of water, all-atom reactant, cosolvent, and oxygens of reactant atoms in each bin

We focus primarily in the `3channel_oxy` representation in this procedure, which was used in the main text.

- Run the script:

  `bash loop_grid_interpolation.sh`

  The script should output the following (only initial parts are shown):

```
(py36_tensorflow2) akchew@swarm:2020_SolventNet_Chem_Sci/scripts $ bash
    loop_grid_interpolation.sh
*** LOADING GENERAL FUNCTIONS (server_general_research_functions.sh) ***
--- Creating database: /home/akchew/scratch/storage/2020_SolventNet_Chem_Sci/
    database/20_20_20_20ns_oxy_3chan_firstwith10 ---
Datatype: All
Solutes: CEL,ETBE,FRU,LGA,PDO,XYL,tBuOH
Temperatures: 403.15,343.15,373.15,403.15,433.15,403.15,363.15
Mass fraction: 10,25,50,75
Database name: 20_20_20_20ns_oxy_3chan
Trajectory location: /home/akchew/scratch/storage/2020_SolventNet_Chem_Sci/
    MD_Simulations/training_set
XTC file: mixed_solv_prod_first_20_ns_centered_with_10ns.xtc
Input dir path: /home/akchew/scratch/storage/2020_SolventNet_Chem_Sci/
    MD_Simulations/training_set/CEL/mdRun_403.15
    _6_nm_CEL_10_WtPercWater_spce_dioxane
```

Listing 5: Outputs of `loop_grid_interpolation.sh`

- Now, you should have the following directories in the `MAIN_dir/database` folder shown in Listing 6. Note that these directories are available in the Zenodo folder by default, without running the `loop_grid_interpolation.sh` script.

```
20_20_20_20ns_oxy_3chan
20_20_20_20ns_oxy_3chan-TESTSET
```

Listing 6: Databases after running `loop_grid_interpolation.sh`

`20_20_20_20ns_oxy_3chan` contains the training data described in Section 2.2, shown in Listing 7. `20_20_20_20ns_oxy_3chan-TESTSET` contains the testing data.

```
CEL_403.15_DIO_10
CEL_403.15_DIO_25
CEL_403.15_DIO_50
...
XYL_403.15_THF_25
XYL_403.15_THF_50
XYL_403.15_THF_75
```

Listing 7: Files within `20_20_20_20ns_oxy_3chan`

These files are pickle files containing histogrammed atomic positions in a frame-by-frame basis. In the main text, we average these frames across 2 ns to generate a voxel representation. As an example of nomenclature, `CEL_403.15_DIO_10` means the pickle contains information of cellobiose in 10 wt% water/90 wt% dioxane at a systemp temperature of 403.15 K. `20_20_20_20ns_oxy_3chan-TESTSET` contain voxel representations for reactants in DMSO, MeCN, and ACE, respectively. Within the `database` directory, there is a folder titled `Experimental_Data`, which contains CSV files with experimental kinetic solvent parameters (*i.e.* labels). Subsequent codes will refer to this directory to obtain labels for training and testing sets.

3.2 Training 3D CNNs

Using the voxel representations, 3D CNNs were trained to predict kinetic solvent parameters (σ), using the following procedure:

- Open the following script:

  `MAIN_dir/scripts/generate_publication_jobs.sh`

- Listing 8 shows the variables that are adjustable in `generate_publication_jobs.sh`. Uncomment by removing the hashtag.

```
## DECLARING TYPES
declare -a run_these_jobs=(\
"MANUSCRIPT_0_TRAINING_5FOLD" \
"MANUSCRIPT_0_TRAINING_3DCNNS_ALLDATA" \
"MANUSCRIPT_1_CROSSVALID_ALLDATA" \
# "SI_0A_Sampling_across_training_size" \
# "SI_0B_Sampling_vs_time_chunks" \
# "SI_0D_200NS_SAMPLING" \
# "SI_2A_Training_different_voxel_inputs" \
# "SI_3A_Training_32_32_32" \
)

## DEFINING MAIN DATABASE TYPE
MAIN_DATABASE_TYPE="20_20_20_20ns_oxy_3chan"
SAMPLING_INPUTS="1.00"
NUM_CROSS_VALID_FOLDS="5"
```

Listing 8: Adjustable variables within `generate_publication_jobs.sh`

The following describes what each job outputs:

- `MANUSCRIPT_0_TRAINING_5FOLD`: Trains 3D CNNs (VoxNet, ORION, and SolventNet) using the 5-fold cross validation procedure used for Figure 4b of the main text.
- `MANUSCRIPT_0_TRAINING_3DCNNS_ALLDATA`: Trains 3D CNNs using all the training data used to predict the test set for Figure 5 of the main text.
- `MANUSCRIPT_1_CROSSVALID_ALLDATA`: Performs leave-one-out cross validation for 3D CNNs, which was used for Figure 6 of the main text.
- `SI_0A_Sampling_across_training_size`: Tests variation simulation time partitions using SolventNet, which was used for Figure S3a in the Supporting Information.
- `SI_0B_Sampling_vs_time_chunks`: Tests partitions of 20 ns to see if there are effects of different partition trajectories, which was used for Figure S3b in the Supporting Information.
- `SI_0D_200NS_SAMPLING`: Trains SolventNet with 200 ns of simulation data, which was used for Figure S3c in the Supporting Information.
- `SI_2A_Training_different_voxel_inputs`: Trains SolventNet with different voxel input representations, which was used to generate Table S4 in the Supporting Information.
- `SI_3A_Training_32_32_32`: Trains SolventNet and VGG16 using $32 \times 32 \times 32$ voxel representations, as discussed for Table S4 and Figure S8 in the Supporting Information.

- Run the bash script to generate training scripts:

  `bash generate_publication_jobs.sh`

  This would create directories within the following directory:

  `MAIN_dir/cnn_output`

  Each output directory is the same as the variable under `run_these_jobs` in Listing 8. The output directories within `cnn_output` is shown in Listing 9.

```
2B_md_descriptor_NN
MANUSCRIPT_0_TRAINING_5FOLD
MANUSCRIPT_0_TRAINING_3DCNNS_ALLDATA
MANUSCRIPT_1_CROSSVALID_ALLDATA_20_20_20_20ns_oxy_3chan_orion_cosolvent
MANUSCRIPT_1_CROSSVALID_ALLDATA_20_20_20_20ns_oxy_3chan_orion_solute
MANUSCRIPT_1_CROSSVALID_ALLDATA_20_20_20_20ns_oxy_3chan_solvent_net_cosolvent
MANUSCRIPT_1_CROSSVALID_ALLDATA_20_20_20_20ns_oxy_3chan_solvent_net_solute
MANUSCRIPT_1_CROSSVALID_ALLDATA_20_20_20_20ns_oxy_3chan_voxnet_cosolvent
MANUSCRIPT_1_CROSSVALID_ALLDATA_20_20_20_20ns_oxy_3chan_voxnet_solute
```

  Listing 9: Files within `cnn_output`

  The description of each folder is shown below:

  - `2B_md_descriptor_NN`: Neural network training for molecular descriptors, which was used to output "Multidescriptor neural network model" in Figure 2d of the main text.
  - `MANUSCRIPT_0_TRAINING_5FOLD`: 5-fold cross validation training procedure used for Figure 4b of the main text. This folder includes training for VoxNet, ORION, and SolventNet.
  - `MANUSCRIPT_0_TRAINING_3DCNNS_ALLDATA`: Training using all training data, which was used to make predictions for Figure 6 of the main text. This folder includes training for VoxNet, ORION, and SolventNet.
  - `MANUSCRIPT_1_CROSSVALID_ALLDATA_20_20_20_20ns_oxy_3chan*`: Leave-one-out cross validation for VoxNet, ORION, and SolventNet across cosolvents and reactants (*i.e.* solute).

  Note that when running `generate_publication_jobs.sh`, it looks for the training set data available within `MAIN_dir/combined_data_set`. This folder contains all a conglomerate of training data using the frame-by-frame voxel snapshots within the `database` folder. This was done because it is computationally more efficient to load one pickle for training rather than 76 individual pickles. All pre-requisite pickles are available in `combined_data_set` to run calculations for the main text. When running `generate_publication_jobs.sh`, it will output jobs to run in a high-performance cluster, available in `MAIN_dir/scripts/job_list.txt`

- Run the training by submitting the `submit.sh` script. Note that this submit script may need to be modified for different servers. This procedure should now allow the user to train 3D CNNs using the dataset described in Section 2.2. All experimental kinetic solvent parameters used for training is available in `MAIN_dir/database/Experimental_Data/solvent_effects_regression_data.csv`. All fully trained networks are available in this repository, described in Section **??**.

- Each directory should output the following (using `20_20_20_20ns_oxy_3chan-split_avg_-nonorm-10-strlearn-1.00-solvent_net-500-CEL_ETBE_FRU_LGA_PDO_XYL_tBuOH-10_25_-50_75-DIO_GVL_THF` within `MANUSCRIPT_0_TRAINING_3DCNNS_ALLDATA` as an example):
  - `extract_deep_cnn.sh`: bash script used to run training
  - `model.chk`: checkpoint file for the model that is stored during training
  - `model.hdf5`: file containing weights associated with the CNN
  - `model.results`: pickle file generated containing post-training information
  - `submit.sh`: submission script that runs `extract_deep_cnn_separated_instances.sh`
  - `model_fold_*.{.chk,.hdf5,.pickle}`: model outputs for 5-fold cross validation (if applicable)

3.3 Scripts used to generate publication images

The script below generates all publication-ready images that were used in the main text: `MAIN_dir/python_scripts/publishable_images.py`

Spyder was used to generate the images. To access Spyder, follow the installation within listing 1 and run the following:

```
# ACTIVATE ENVIRONMENT
conda activate py36_tensorflow2
# TO ACTIVATE SPYDER:
spyder --new-instance
```

Listing 10: Instructions to open up Spyder, which was used to develop the publication images.

Before running the scripts, change the global path within: `MAIN_dir/python_scripts/core/global_vars.py`

Change the `PATH_MAIN_PROJECT` variable to the main path where you stored the Zenodo output folder, as shown in Listing 11.

```
## DEFINING PATH TO MAIN DATA
try:
  PATH_MAIN_PROJECT = check_path_to_server(r"/Volumes/akchew/scratch/storage/2020
    _SolventNet_Chem_Sci/")
except Exception:
  PATH_MAIN_PROJECT = ""
  pass
```

Listing 11: Main path to the project that needs to be changed within `global_vars.py`.

Then, open `publishable_images.py` and run each section one-by-one using CTRL + Enter for Windows or Command + Enter on the Mac. Sections are divided by the `%##` symbol. In the Figures below, it shows you the output of each section required to re-make the images of the main text. To save the images, change the `save_fig` variable to "True" and edit the `path_image_dir` to the path of output. For each Figure, we have saved them as an *.svg file, then currated the image using Adobe Illustrator.

```python
#%%
## MAIN FUNCTION
if __name__ == "__main__":

    ######################################################################
    ### FIGURE 2C -- Human descriptors
    ######################################################################

    ## DEFINING FIGURE NAME
    fig_name = r"2C_Multilinear_regression"

    ## DEFINING FIGURE SIZE
    figure_size=( 7.6, 7.6 )

    ## DEFINING INPUTS
    inputs={
            'path_md_descriptors': path_dict['path_md_descriptors'],
            'molecular_descriptors' : [ 'gamma', 'tau', 'delta' ],
            'output_label' : 'sigma_label',
            'verbose' : False,
           }

    ## LOADING DESCRIPTORS
    analyzed_descriptors = analyze_descriptor_approach(**inputs)

#    ## RUNNING PER COSOLVENT TRAINING
#    model_storage = analyzed_descriptors.generate_multilinear_regression(
#                                    analyze_type = 'all',
#                                    # analyze_type = 'cosolvent',
#                                    normalize=True,
#                                    verbose = True,
#                                    )

    ## GETTING CROSS VALIDATION
    n_folds = NUM_CROSS_VALIDATION_FOLDS

    ## N FOLD CROSS VALIDATION MODELS
    cross_valid_model_storage, indices_dict = analyzed_descriptors.cross_valid_generate_n_folds(n_folds = n_folds)

    ## GETTING TEST SET FOR EAHH FOLD
    test_set_each_fold = analyzed_descriptors.predict_test_set_for_n_folds(cross_valid_model_storage = cross_valid_model_storage,
                                                                           indices_dict = indices_dict,
                                                                           pred_type = "each_fold")

    ## PLOTTING
    plot_parity_publication_single_solvent_system( dataframe = test_set_each_fold,
                                                   fig_name = os.path.join(path_image_dir, fig_name) + '.' + fig_extension,
                                                   mass_frac_water_label = 'mass_frac_water',
                                                   sigma_act_label = 'sigma_label',
                                                   sigma_pred_label = 'sigma_label_pred',
                                                   fig_size_cm = figure_size, # (16.8/3, 16.8/3),
                                                   fig_extension = fig_extension, # fig_extension
                                                   save_fig = save_fig)
```

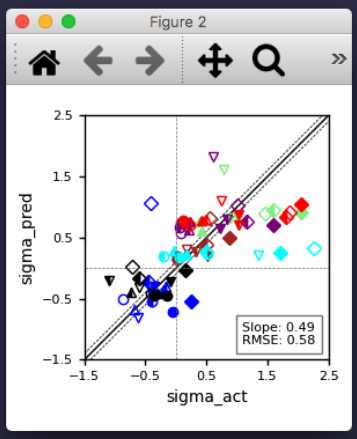

Figure 2: Running `publishable_images.py` to generate Figure 2c of the main text.

Figure 3: Running `publishable_images.py` to generate Figure 2d of the main text.

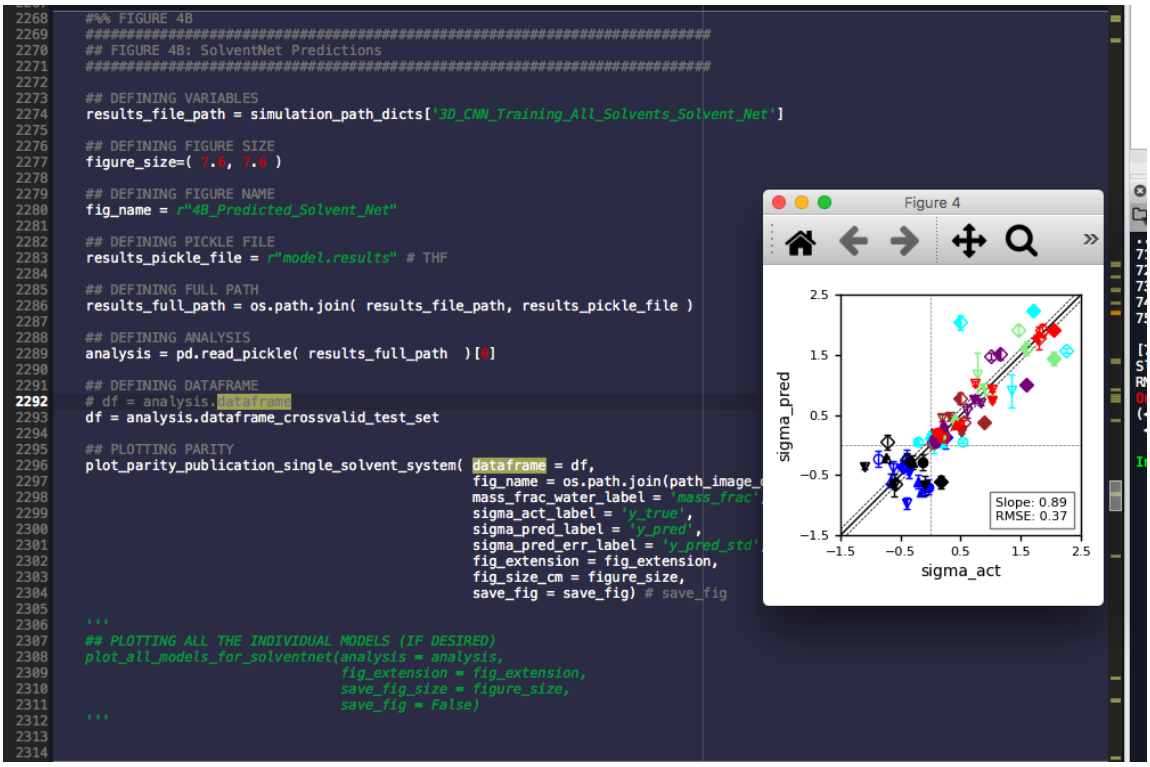

Figure 4: Running `publishable_images.py` to generate Figure 4b of the main text.

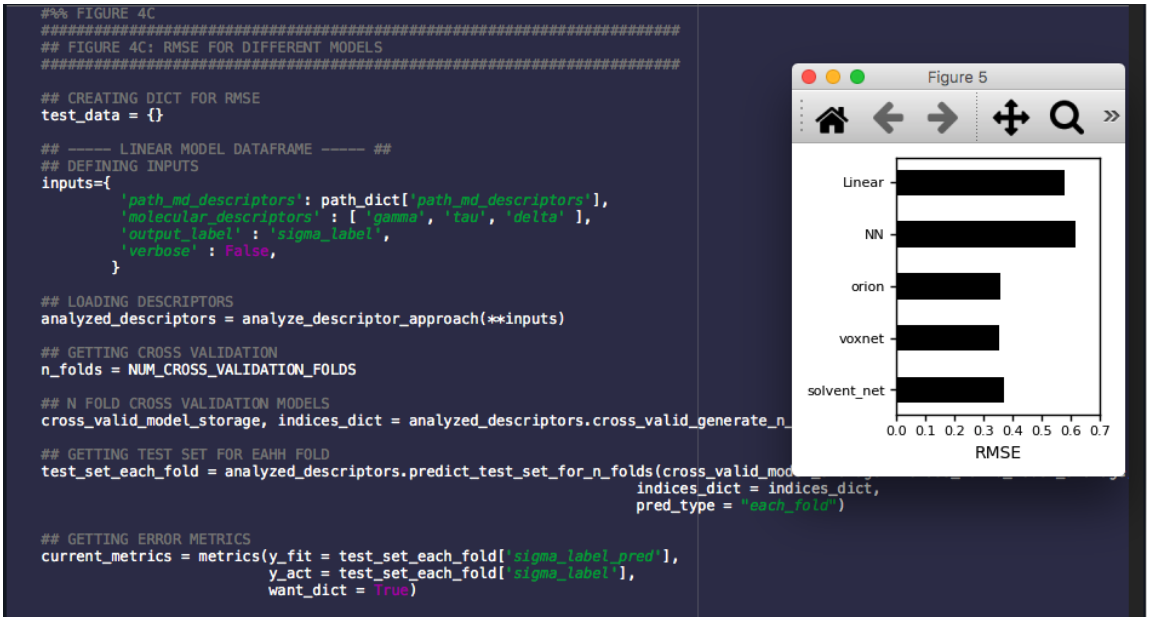

Figure 5: Running `publishable_images.py` to generate Figure 4c of the main text.

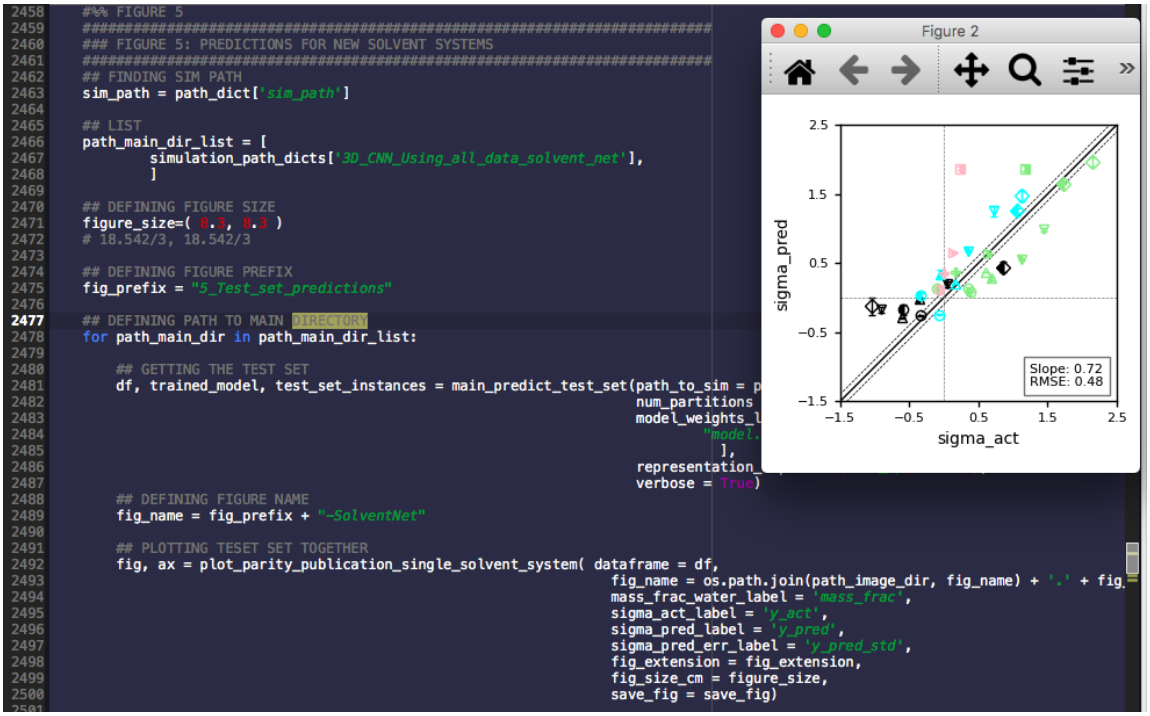

Figure 6: Running `publishable_images.py` to generate Figure 5 of the main text.

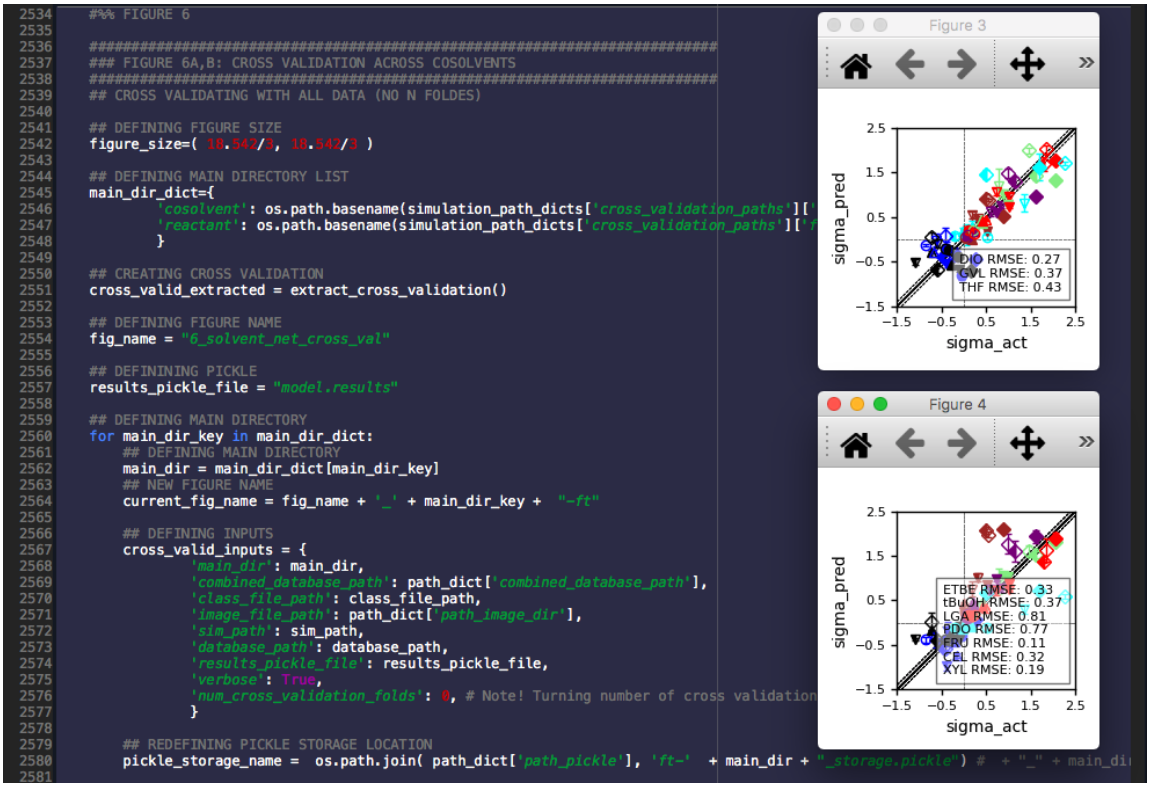

Figure 7: Running `publishable_images.py` to generate Figure 6 of the main text.

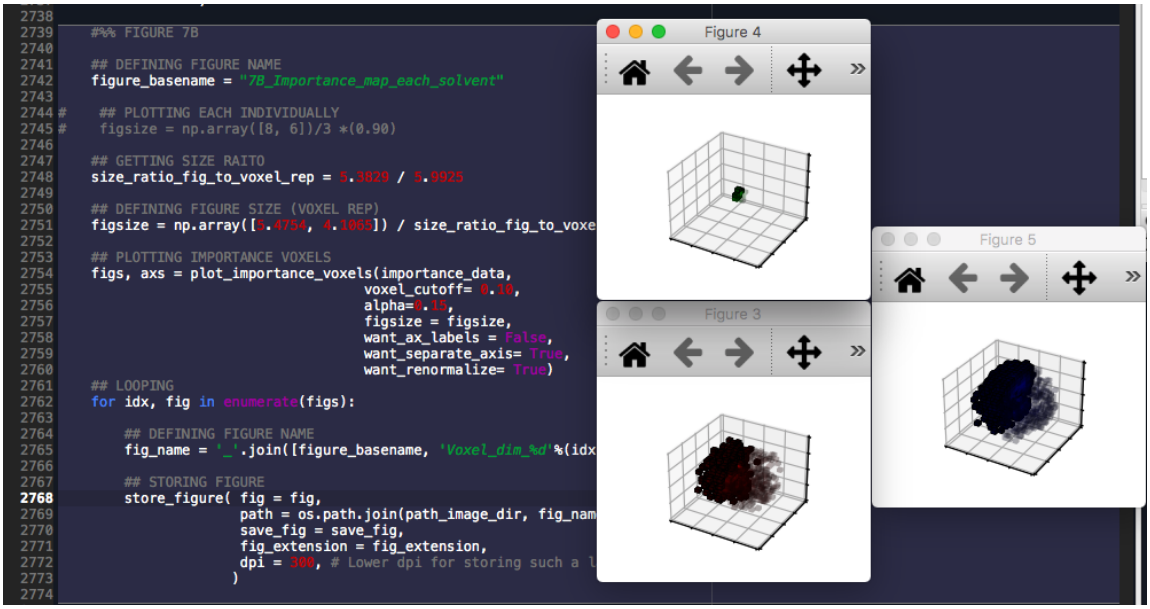

Figure 8: Running `publishable_images.py` to generate Figure 7 (top) of the main text.

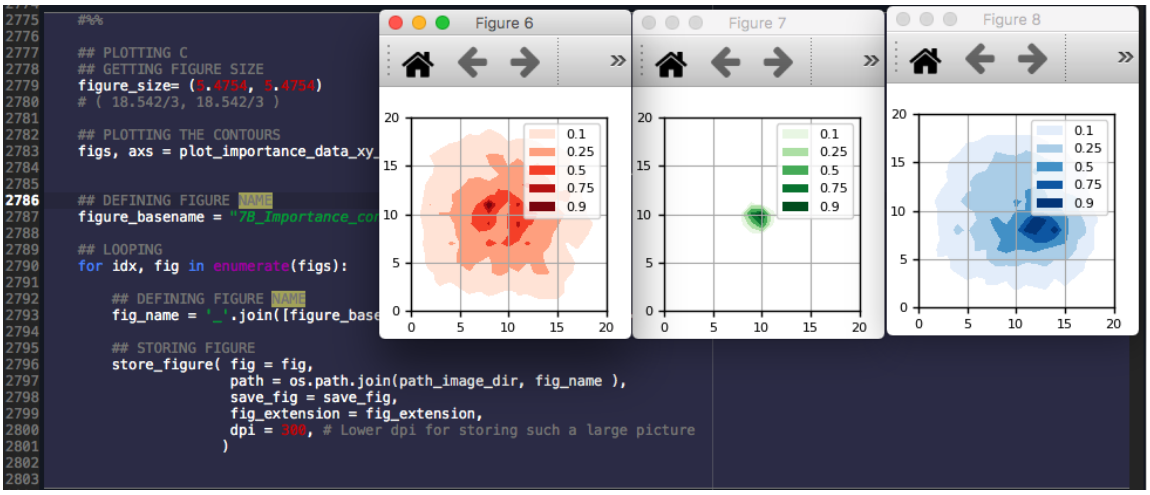

Figure 9: Running `publishable_images.py` to generate Figure 7 (bottom) of the main text.

At this point, you should be able to generate all the images in the main text. Data used to generate the Supporting Information is not available in this database to limit the data size. Data for Supporting Information is available upon request and also could be generated from Section 3.2.

4 Appendix

4.1 Accessing 3D CNN models

All 3D CNN models are available within:
`MAIN_dir/python_scripts/`

- VoxNet: `deep_cnn_vox_net.py`
- ORION: `deep_cnn_ORION.py`
- SolventNet: `deep_cnn_solvent_net_3.py`

Please feel free to import the CNN models from this script and try them out!

4.2 Accessing Tensorflow-GPU for faster training

We leveraged a high-throughput cluster at UW-Madison with 1 GPU / 1 core to quickly perform calculations. If you have GPUs available, consider installing Tensorflow-GPU:
`pip install tensorflow-gpu -user`

We also have some code to transfer the simulations from CPU-only jobs to GPU-enabled jobs for high-throughput clusters. To do this:

- Go to `MAIN_dir/scripts/`.
- Run the following: `bash switch_submit_deep_cnn_chtc.sh FOLDER_NAME "" true`
  Change the `FOLDER_NAME` into the folder within the `MAIN_dir/cnn_output` folder. After running this script, it should output a new submit script, which is for a high-throughput cluster. This works for UW-Madison high-throughput cluster, so the code may need to be modified for different clusters.

References

[1] Theodore W Walker, Alex K Chew, Huixiang Li, Benginur Demir, Z Conrad Zhang, George W Huber, Reid C Van Lehn, and James A Dumesic. Universal kinetic solvent effects in acid-catalyzed reactions of biomass-derived oxygenates. *Energy & Environmental Science*, 11(3):617–628, 2018.