* Please submit your manuscript to <https://www.editorialmanager.com/star-protocols/default.aspx>. Be sure to upload the Word version of this manuscript at submission.

Implementation of the Polyfingerprint a molecular structure representation into an artificial neural network.

**Yannik Köster,1,2,3,\* Julian Kimmig,1,2 Stefan Zechel,1,2 Ulrich S. Schubert1,2,4,\*\***

1 Laboratory of Organic and Macromolecular Chemistry (IOMC), Friedrich Schiller University Jena,

Humboldstr. 10, 07743 Jena, Germany

2 Jena Center for Soft Matter (JCSM), Friedrich Schiller University Jena,

Philosophenweg 7, 07743 Jena, Germany

3 Technical Contact

4 Lead contact

\* Correspondence: yannik.koester@uni-jena.de

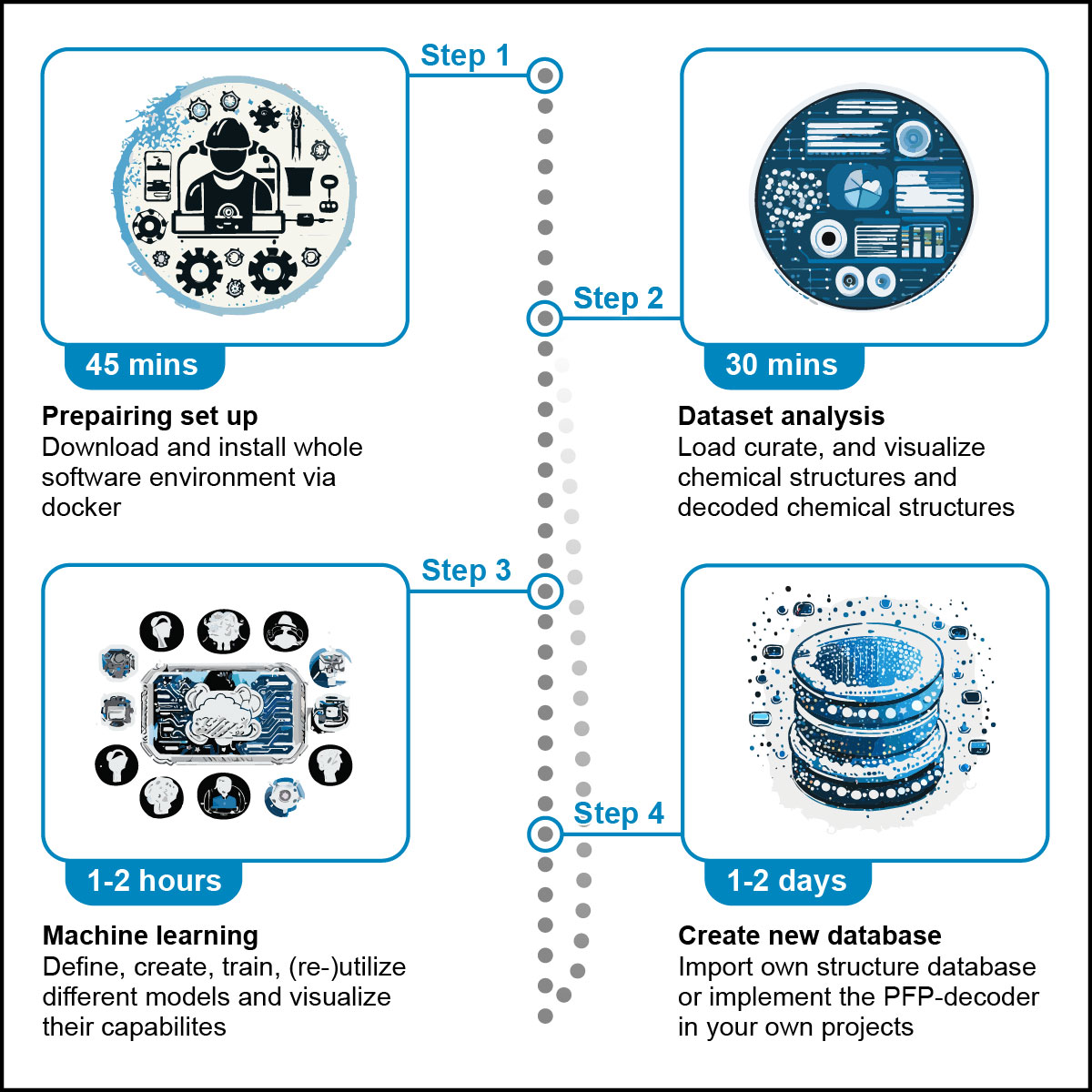
\*\* Correspondence: ulrich.schubert@uni-jena.de

# Summary

To represent polymeric chemical structures succinctly to be available for computational tasks, especially for machine learning tasks, a decoder is needed. This protocol explains how to use a polymer specific decoder which creates a representation/fingerprint called the Polymerfingerprint (PFP). The Protocol is designed in such a way that no knowledge of programming is necessary. Furthermore, it will be explained how this Fingerprint integrates within an artificial neural network to predict properties of polymers based off their structure only.

**For complete details on the research behind this protocol, please refer to Köster et al., 2023** (<https://doi.org/10.1016/j.xcrp.2023.101553>)

# Graphical abstract



# Before you begin

The protocol below describes how to set up and run a polymer property prediction model, more precisely, one for cloud points. However, this platform can also be used to predict any polymer property, based on the chosen input.

**Timing: 10 min.**

## Hardware requirements:

A 64-bit processor with CPU support for virtualization.

Further in-detail requirements can be read up the official docker docs under the operating system specific site at https://docs.docker.com/desktop/.

## Exemplary hardware requirements:

Minimal Hardware requirements for using the trained model:

Operating system: Windows, Linux, iOS. (tested on Windows 10, ¿Ubuntu 20.04?, Mac Sonoma 14.1 with Intel Processor)

At least 4 GB of RAM

20 GB of free disk space

Intel Core i5 4x2.60 GHz

Recommended hardware for training a new model:

Operating system: Windows Vista SP2, Linux Ubuntu 20.04 or newer.

16 GB of RAM

25 GB of free disk space

Ryzen7 8x3.7 GHz processor or dedicated graphics card from Nvidia (4 GB)

## Installation of the development environment:

**Timing: 45 min.**

1. Download the program, training data and models from Github at https://github.com/Bizbalt/PFP/archive/refs/heads/main.zip and unpack it to a location of your choice (from now on referred as <PFPDIR>).

**Optional:** If you got a Nvidia graphic processing unit you can uncomment the last codeblock in the docker-compose.yml file in the newly created PFP folder to make it available to the project.

\PFP\docker-compose.yml:

to

># uncomment to use GPU

>#deploy:

># resources:

># reservations:

># devices:

># - driver: nvidia

># capabilities: [gpu]

># uncomment to use GPU

>deploy:

> resources:

> reservations:

> devices:

> - driver: nvidia

> capabilities: [gpu]

1. Download and install Docker for your operating system at docker.com/get-started
   1. Mac:

**Note:** Make sure your downloaded Docker Version matches the chip-architecture of your Computer (standard processor chip was Intel but can be Apple. You can check for your chipset under “About this Mac” in the Apple menu.

* + 1. Double click the docker.dmg and drag it into the applications folder.
    2. Open the App and accept the terms and conditions. Let it configurate.
    3. Open a terminal (Applications > Utilities > Console) in admin mode, or with the “sudo “ prefix and run the following commands:  
       >cd <PFPDIR>

**Note:** You can check what files your current path contains with  
>sudo pwd

* + 1. Install the project environment via docker.

>docker compose up

**Note:** If you are experienced with installation or want to install the software over ssh you can install the docker services from the release channels <https://docs.docker.com/engine/install/>.

* 1. Windows:
     1. Run it with normal permission (no admin rights needed).
     2. Tick “use Windows Subsystem for Linux 2 (WSL2) instead of Hyper-V” if applicable.
     3. Close and restart system like instructed.
     4. Accept the terms and conditions on the pop-up after start-up and close it.
     5. Open the windows command line.
     6. (Win + r) Write “cmd” and hit enter.
     7. In the console navigate to the project folder with the change directory command.

>cd <PFPDIR>

* + 1. Install the project environment via docker.

>docker compose up

**Hint:** When prompted to install git in the terminal accept and agree to license.

* 1. Linux:
     1. Follow the recommended installation instructions for your distribution of Linux as mentioned on <https://docs.docker.com/engine/install/>.
     2. Navigate to the project folder containing the docker image:

>cd <PFPDIR>

* + 1. Install the project environment via docker.

>sudo docker compose up

1. The Jupyter server starts automatically after the installation process is complete. It can be access it via your browser of choice at

“http:localhost:8888”

1. And pass in the password “pfppassword”.

**Pause point**: Everything is now set to run the software and to run it only two commands and step 3 must be redone after a shutdown:

>cd <PFPDIR>

>docker compose up

# Key resources table

|  |  |  |
| --- | --- | --- |
| REAGENT or RESOURCE | SOURCE | IDENTIFIER |
| Deposited data | | |
| Non curated LCST dataset | Köster et al. 2023 | Non-curated Dataset.xlsx |
| Curated LCST dataset | Köster et al. 2023 | cloud\_points\_data.csv |
| Software and algorithms | | |
| Anaconda | Anaconda, Inc. | https://www.anaconda.com |
| ChemDraw Professional 21.0.0.28\* | PerkinElmer | <https://www.perkinelmer.com/category/chemdraw> |
| Explanatory Jupyter notebook | Köster et al. 2023 | Cloudpoint determination.ipynb |
| Cloud point prediction network | Köster et al. 2023 | cloudpointprediction.py |
| Polyfingerprint decoder | Köster et al. 2023 | Polyfingerprints.py |
| Other | | |
| Cheminfo SMILES checker | University of Valle | http://www.cheminfo.org/flavor/malaria/Utilities/SMILES\_generator\_\_\_checker/index.html |

\*Or any other structure drawing software, that allows the export and import of chemical structures as SMILES

# Step-by-step method details

## Data curation and analysis Jupyter-notebook

**Timing: 30 min.**

**Note**: You could start at this point after a restart. [Troubleshooting 1](#_Problem_1:_[State), [Troubleshooting 2](#_Problem_2:_1)

1. With Jupyter being open in the browser, open the “examples/cloud\_point” folder in the folder tree on the left and then double click the “data\_curation\_and analysis.ipynb” file. This is one of the three Jupyter notebooks where code can be executed and examined sequentially.

For good measure every line including an import statement and every code block should have its own cell (add another via the plus or Shift + Enter). You do not need to type in the code just execute it separately as all of it is stored in the Jupyter notebook already. You can pause at any point.

**Note**: The cells also store the previous output. Even though: When you get back to it make sure the cells above the one you want to execute are all executed in succession before as they are dependent building up on each other and their output will just then be stored in your RAM again.

**Note**: You can always change some parameters in a cell and then run/try out this variation to get a better understanding of the program.

**Optional**: You can use your own dataset as described under the [Optional: prepare own dataset](#_Optional:_prepare_own) section, but it is recommended to follow the Tutorial with the LCST data first.

1. This first notebook is meant to check and prepare the dataset for the training in the next notebook.

**Note:** Besides the explanation here, the code sometimes contains comments which work as headings or further technical elucidation.

># comments like this one are marked with a hashtag and increase code readability

You can add comments at the end of every line as you like.

1. The first cell withholds import statements for libraries we need to load and manipulate our chemical structure data.

**Note:** After placing the curser somewhere in the cell and pressing shift+enter it executes and jumps to the next cell.

1. The next two cells hold declaration of our working directory and the loading of it.
2. The fourth cell contains the data-specific curation of our “lower critical solution temperature” dataset and uses the datareader from the pfp library to bring the tabular data to programmatically better manageable form while declaring which columns to use for training. The reader automatically prepares the columns for later training. Lastly the information is stored as an infofile.
3. The next six cells create diagrams to relate the chemical structures to each other, which citation they origin from and so on. These are also specific to our “lower critical solution temperature” dataset and function as exemplary analysis.

**Note:** On default the figure.show()lines for the interactive plots are commented out to save RAM. Uncommenting and rerunning the cell displays them.

1. All further cells cover chemical structure representation.
   1. How to display the SMILES as graphics
   2. How to use the get just the fingerprint vectors and an excursus to comparability

**Note:** General fingerprints are well explained by the [daylight side](https://www.daylight.com/dayhtml/doc/theory/theory.finger.html). For more in detail information about the PFP read the [original article](https://doi.org/10.1016/j.xcrp.2023.101553).

* 1. Creating different PFP set sizes and
  2. Displaying them in two cells in two forms.
  3. Calculating conventional similarity indexes, coefficients, distances etc.

## Train and use prediction model Jupyter-notebook

1. Open the “train\_and\_use\_prediciton\_model.ipynb” Jupyter notebook next.
2. As in the last Jupyter notebook the first cell imports program libraries that we need to load, train with, save and use our previously curated data set and the corresponding info file.
3. The following two set the working directory and load the curated data set back into the RAM.

**Note:** This needs to be done for this notebook again, as Jupyter notebooks do not share variables, even if both were running. This also means that that the “data\_curation\_and analysis.ipynb” notebook can be closed to save resources.

1. The fourth cell sets the PFP-hyperparameters, creates and reduces the PFP. At the last line the reduction data is stored exclusively.

**Critical: A model is representation-specific. This means that any new data that shall pass through the model must be represented and reduced in exactly the same way as previous training data. For this reason, the reduction data is saved.**

1. The next two cells just visualize the PFP datatype and how to access its train and goal data

**Note:** Jupyter cells output the last variable even without print() or display() functions

1. The second to last cell sets individual identification and hyperparameters for the model to be created and trained.
2. The last cell contains mainly the train\_model() function. The training is performed based on the hyperparameters previously defined. If the model with the specified name already exists it loads the model and continues the training where it was stopped in the last run.

**Critical: Here it is important that the same hyperparameters are used, if they are changed, the hyperparameters of the loaded model and the defined ones can mismatch which would result in an error. In such cases we recommend to either delete the model folder completely or simply choose a new model name.**

## Inference Jupyter-notebook

1. Open the “inference.ipynb” Jupyter notebook.
2. This third notebook is built for the utilization of an already trained model.
3. In the same manner as the notebooks before the first three cells import the same libraries, set the path and name to the desired model and load a dataset supposedly missing the target value.
4. In the fourth cell the PFP is created, and additional training information is set and fitted to the chosen model.
5. In the subsequent cell the PFP is reduced model specific (see step 11).
6. The model is loaded in cell six.
7. In cell seven the target variable is calculated by the model based on the structural PFP data as input.
8. Lastly the calculated data is inserted into the initial data set and displayed.

## Optional: prepare own dataset

**Timing: 1-2 days (depending dataset size).**

**Note:** The PFP was designed to also work with small datasets. 500 Entries are considered small.

1. The probably easiest way to implement your own data is by making a copy of the template folder and renaming the copy to a descriptive name for your data.
2. In the newly generated folder is a excel file “excel\_template.xlsx” which can be used as a starting point to fill in your own data.
3. The structure is as follows:
   1. As many pairs of ‘SMILES\_repeating\_unit\_#’ and ‘molpercent\_repeating\_unit\_#’ columns need to be created as the polymer with the most different monomers has monomers. The # must be unique in the table, we recommend increasing numbering or letters (1,2,3… or A,B,C,…). The “SMILES\_repeating\_unit\_#” contains valid SMILES for the repeating unit, which have to start and end with an atom with incomplete valency (radical).

**Critical: It is crucial that the connection points of the repeating units are written explicitly,** like radicals ( [CH2]C[CH2] ) Ein Bild, das Screenshot, Text, Diagramm, Reihe enthält.

Automatisch generierte Beschreibung

All sidechains and/or backbone atoms must be placed in between the two radical groups or appended in brackets, e.g. [CH2]CC[CH2] and [CH2]C[CH](C) are valid but [CH2]C[CH]C is not. The reason for that is the internally concatenation of the SMILES strings, e.g.

[CH2]CC[CH2] becomes

…[CH2]CC[CH2][CH2]CC[CH2][CH2]CC[CH2]… which is a valid polymer but

[CH2]C[CH]C becomes

…[CH2]C[CH]C[CH2]C[CH]C[CH2]C[CH]C… which is structurally also valid but is not correlating to the intended repeating unit with a -CH3-side group in the polymer.

‘molpercent\_repeating\_unit\_#’ should contain the amount of the specific repeating unit in the polymer and the values should either be absolute or in percentage. Each polymer must contain at least on SMILES-molpercentage pair, but it is not necessary to fill all available columns, since empty ones will be ignored.

* 1. Similar to the repeating units start- and end-groups are defined in ’SMILES\_start\_group’ and ‘SMILES\_end\_group’. Here the startgroup has to end with a radical to be prepended to the repeating unit string and in the same manner the end-group SMILES have to start with a radical to be appended.
  2. Also mandatory is the ‘Mn’ column which contains the number average total masses of the polymers (other masses like Mw or even the degree of polymer could also work, but Mn has the most descriptive value).
  3. Additives often have a significant influence on the polymer properties, which is why we implemented them as well. Similar to the repeating units two columns are necessary for each additive: ‘additive\_#’ and ‘additive\_#\_weight\_percent’ or ‘additive\_#’ and ‘additive\_#\_concentration\_molar’ where # is again a unique identifier. If you use ‘additive\_#\_weight\_percent’ the respective concentration will be calculated internally using the density property from the ‘data’ sheet in the excel file. ‘additive\_#’ again must be valid SMILES, but in this case it can be arbitrary smiles. Even salts in the form of [Na+].[Cl-] are possible where the individual ion/molecules, separated by ‘.’ are threaded as individual additives. If you have additives where no SMILES structures can be given e.g., a certain amount of a complex mixture, we recommend creating a new numerical column for this additive.
  4. All other columns are treated either as numerical or categorical values. If the column contains numbers, it is treated as a numerical column and will add one extra dimension to the training data. Columns which cannot be parsed as numerical, e.g., strings are interpreted as categorical: each unique value in this column adds a new dimension to the training data, which is either 1 if it is in the specific formulation or 0 if it is not (one-hot-encoding). A column with the values A, B, C, A and C, would add 3 dimensions to each entry, with the values [1,0,0] for the “A” [0,1,0] for the “C” and so on.

**Table 1. Example of information of an entry for a RAFT copolymerization of Vinyl Acetate and *N*-Vinylcaprolactam (line 434 in the *cloud\_points\_data.csv* file from Etchenausia *et al.*** [**https://pubs.acs.org/doi/10.1021/acs.macromol.6b01451**](https://pubs.acs.org/doi/10.1021/acs.macromol.6b01451)**). Transformed to a long format for better readability.**

|  |  |
| --- | --- |
| SMILES\_start\_group | [C](C)(C)(C#N) |
| SMILES\_end\_group | [S]C(=S)OCC |
| SMILES\_repeating\_unitA | [CH2][CH1](OC(=O)C) |
| molpercent\_repeating\_unitA | 0,53 |
| SMILES\_repeating\_unitB | [CH2][CH](N1C(=O)CCCCC1) |
| molpercent\_repeating\_unitB | 0,47 |
| Mn | 35370 |
| PDI | 1,1 |
| polymer\_concentration\_wpercent | 0,003 |
| cloud\_point | 21,4 |
| def\_type | C |

**Note**: As the representation of Structures as SMILES codes can be tricky, one can use the [SMILES generator/checker from the Universtiy of Valle](http://www.cheminfo.org/flavor/malaria/Utilities/SMILES_generator___checker/index.html) or use the *paste special* (Alt+Ctrl+p) and *copy as* (Alt+Ctrl+c) SMILES functions of ChemDraw to verify the codes. The third *Namen* tab of the *Non-curated Dataset.xlsx* file also gives many examples for initiators and repeating units.

**Optional**: The default hyperparameters can also be seen and set manually through the hyperparameter.yml files created.

1. Similar to the lower critical solution/cloud point temperature example notebooks, the “training” and “inference” notebooks have to be run with tuned parameters. Especially the TARGET\_VALUE has to be set to the specific column name of your output value.

## How to use the Polyfingerprint library

**Timing: 10 min.**

**Note:** All functions for the decoder are stored in polyfingerprints.py file

1. The crucial function is create\_pfp(). When we take the same example from above it takes the structure and composition of the polymer or list of polymers in the following way.

>import polyfingerprints as pfp

>a\_polyfingerprint = pfp.create\_pfp(end\_units={"start": "[C](C)(C)(C#N)", "end": "[S]C(=S)OCC"}, repeating\_units={0.53 :"[CH2][CH1](OC(=O)C)", 0.47: "[CH2][CH](N1C(=O)CCCCC1)"}, mol\_weight=35370, fp\_size=2048)

>print(a\_polyfingerprint)

Since the dictionaries for the repeating units can be arbitrary long, copolymers with arbitrary many different repetition units can be set in here.

**Note**: More options to all functions can be seen opening the Contextual Help (Ctrl + I).

1. To shorten training times a list of fingerprints can be reduced with the reduce\_fp\_set() function. The function checks if some parts of the fingerprints do not differ and throws the obsolete out.

>b\_polyfingerprint = pfp.create\_pfp(end\_units={"start": "[C](C=C)(C)(C#N)", "end": "[S]C(=S)OCCCC"}, repeating\_units={0.53 :"[CH2][CH1](OC(=O)CC)", 0.47: "[CH2][CH](N1C(=O)CCC1)"}, mol\_weight=35370, fp\_size=2048)

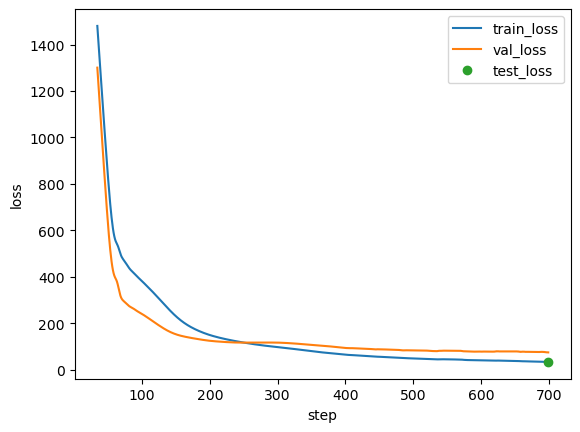
>list\_of\_pfps = [a\_polyfingerprint, b\_polyfingerprint]

>print(pfp. reduce\_fp\_set(\*list\_of\_pfps))

**Note**: Notice that that the function also gives out two masks. These masks can be used later to reduce a same sized (*eg.* fp\_size=2048) fingerprint from polymer outside of the set with the reduce\_fp() function in the same way. Like this they will be still comparable. The reduce\_fp() function will also give out a loss, which describes how much of the information that was cut off was not obsolete.

# Expected outcomes

Though the outcome of the training is random, nearly every case of example training with the default parameters should be in between 20 and 40 °C² mean squared error. The console will just output that number as “loss” and print a graphic like the following.



**Figure 1. Training overview cross-plot generated after a default run.**

All further expected outcomes are similar to the outputs of the default example notebooks.

# Quantification and statistical analysis (optional)

Table 1 below provides an idea of how arbitrarily fast the training process can occur can be. If changes to the hyperparameters or the data .csv file are made more than one in ten trainings may not result in a reasonable error in a reasonable time.

**Table 1. Test for time and speed of training. The training was stopped after 10 Epochs of training. These early training stages are a good forecast for the overall training runs. Default Hyperparameters are used.**

|  |  |
| --- | --- |
| **Time in Seconds** | **Temp. MSE in °C²** |
| 29 | 864 |
| 26 | 895 |
| 29 | 757 |
| 30 | 824 |
| 27 | 507 |
| 26 | 913 |
| 29 | 2061 |
| 25 | 893 |
| 25 | 602 |
| 27 | 977 |

# Limitations

When adding training data, it must always be kept in mind that single entries of a completely different subject area which is hardly brought in comparison with the other entries is likewise predicted poorly. As all entries are split into the training, validation and test sets (in an 8:1:1 ratio) it is recommended to have at least 10 entries for each subject area. As the program also outputs a csv with all predicted temperatures in the end of training, one can easily see the performance on specific entries there.

## Deinstallation:

If you want to deinstall the software, deinstall the docker PFP environment (e.g., in docker desktop or via the console), before you deinstall docker. [Troubleshooting](#_Problem_4:) 3

# Troubleshooting

## Problem 1:

The docker compose up command does not work.

>ERROR: error during connect: […]

## Potential solution:

* Open “docker desktop” from your programs before executing the command again. Your system might not be able to run the docker engine automatically.

## Problem 2:

The Notebook does not run anymore or you just want to revert the original state.

## Potential solution:

* You can delete the examples folder as it will be reinstated on next startup/compose.
* Alternatively, you can also rebuild the whole environment with

>docker compose up –-build

## Problem 3:

Deinstallation seems incomplete, since hard drive space is still less than before.

## Potential solution:

* Did you deinstall the environment in the Docker software before you deinstalled Docker itself?
  + Otherwise reinstall and deinstall.
* Open your Local folder by opening the windows search for

>%localappdata%

* + Delete the docker folder there manually.

# Resource availability

***Lead contact***

Further information and requests for resources and reagents should be directed to and will be fulfilled by the lead contact, Ulrich S. Schubert (ulrich.schubert@uni-jena.de)

***Materials availability***

This study did not generate unique reagents.

***Data and code availability***

The dataset and code used during this study are available at Github: <https://github.com/Bizbalt/PFP>. The primary research and the dataset on which this study is based on is available at https://doi.org/10.1016/j.xcrp.2023.101553.

# Acknowledgments

[Enter the following information here: 1) all funding sources; 2) collaborators and/or core facilities contributing to the work.]

# Author contributions

Conceptualization, methodology, software, validation, investigation, Y.K., J.K.; resources, U.S.S.; writing – original draft, Y.K.; writing – review & editing, U.S.S., S.Z., and J.K.; funding acquisition, U.S.S.; supervision, U.S.S. and S.Z.

# Declaration of interests

# The authors declare no competing interests.

# [Please disclose competing interests for all submitted content by completing and submitting the [“declaration of interests” form](http://www.cell.com/pb/assets/raw/shared/forms/di_form.pdf).]

# References

Köster, Y., Kimmig, J., Zechel, S., and Schubert, U.S. (2023). Fingerprint applicable for machine learning tested on LCST behavior of polymers. Cell Rep. Phys. Sci. *4,* 101553. 10.1016/j.xcrp.2023.101553.

Etchenausia, L., Rodrigues, A.M., Harrisson, S., Deniau Lejeune, E., and Save, M. (2016). RAFT Copolymerization of Vinyl Acetate and N -Vinylcaprolactam: Kinetics, Control, Copolymer Composition, and Thermoresponsive Self-Assembly. Macromol. *49,* 6799–6809. 10.1021/acs.macromol.6b01451.

# Figure legends

Figure 1: [Include a clear and concise figure title for each figure. Figure legends are suggested, but not required.]