-This file describes all the options used to train the symmetry conforming neural network for predicting diffusion relaxation vectors using the GCNetRun.py module in the Symm_Network directory.

Pre-requisites:
-Running the networks always requires the path to a dataset file. The previous examples in the directory "1_Binary_Lattice_Gas_data_Generation" have Jupyter notebooks that show how to generate these dataset for lattice gas type of simulations. Later, examples in directories "4_Monte_Carlo_LAMMPS" and "5_Kinetic_Monte_Carlo_LAMMPS" show how to generate such datasets that can be used by the training code using vthe MEAM potential of Choi et.al. (2018) to generate datasets for the 5-component Cantor alloy.

-We also need to specify a path to a crystal data file that contains and supercell and symmetry-related information. Since in this repository we simulate only FCC-based alloys, such an example crystal data file can be found in the "CrysDat_FCC" directory in our repository home. Note that to generated the datasets and train/evaluate the networks, the same crystal data file must be used, since it contains the group operations and jump directions in a specified order.

-The example job scripts provided describe typical settings in which the neural networks are run.

-The networks are run in 1 of 3 main “modes” (--Mode option).

- The "train" mode: This is for training networks.
  - When the networks are trained from scratch on a fresh dataset, the --Scratch (-scr) flag must be set.
  - All trained networks are saved to directories with a name of the form "ep_T_{0}_{1}_n{2}c{3}_all_{4}", where:
    - {0} : Temperature (or composition for the binary system) specified by the --TNet argument.
    - {1} : Either the string "vac" or the integer label of the atomic species whose transport coefficient is optimized. For example, in the binary lattice gas simulations, the fast species has label "1" while the slow species has label "0".
    - {2} : The no. of intermediate layers (excluding input and output layers) in the neural network. Note that the input layer always has as many input channels as the no. of atomic species, and the output layer has (typically) only one channel (although more channels can be specified).
    - {3} : The no. of input/output channels of the intermediate layers. For simplicity, all such layers have the same input/output channels.
    - {4} : For all the example runs, as well as the runs in our data base this is "0". If the -aj (--AllJumps) option is set, this value is "1". The -aj option corresponds to considering all jumps out of a state, instead of a single KMC step jump out of the state, and can lead to significant increase in training time, which is why the -aj flag is typically not used during training.
  - The -sep (--Start_epoch) option gives the first epoch to start training from while the –eep (--End_epoch) option gives the last epoch. All such networks are saved at intervals given by the -i (–Interval) option. If the –sep option is greater than zero, a network saved at the epoch must be in the network directory mentioned above.
  - All networks are saved as PyTorch state dictionaries with the name “ep_{0}.pt”, where {0} is the epoch (an integer).

- The “eval” mode:
  - This mode is used to load the network from the saved directory for a particular training run, and to compute the transport coefficients. The -sep (--Start_epoch) option gives the first epoch to consider while the –eep (--End_epoch) option gives the last epoch. All such networks are evaluated at interval given by the -i (--Interval) option is used.
  - The training and validation set transport coefficients are saved in “.npy” files for each epoch in files with names of the form “tr_{0}_{1}_{2}_n{3}c{4}_all_{5}.npy” and “val_{0}_{1}_{2}_n{3}c{4}_all_{5}.npy” in the working directory, where fields {0}, {2}, {3}, {4} and {5} have the same meaning as the names of the save directories of the networks discussed before. The option {1} is given by the -tn (--Tdata) option, and corresponds to the temperature (or composition for the binary system) from which the data was read. Note that {2} is given by the -tn (--Tnet) option which specifies the temperature (or composition) to whose data the networks were trained, and networks are loaded from that corresponding directory. This to enable transfer learning. For example, we can train networks at 1373 K, and then evaluate transport coefficients with these networks on a data set constructed at 1073 K, in which case, we specify -tn 1373 -td 1073 in our training run.

- The “getY” mode:
  - This mode is used to load a network at a particular epoch and compute relaxation vectors of a state.
  - These relaxation vectors are also saved numpy binary “.npy” files with names that are of the form “y_st1_{0}_{1}_{2}_n{3}c{4}_all_{5}_{6}.npy” for the initial states of a vacancy jump, where the first five fields have the same meaning as discussed in the “eval” mode above for the training and validation set transport coefficient files. The last field {6} is an integer corresponding to the epoch at which the saved network was loaded.
  - For the final state of a vacancy jump in our dataset files, this file will have the name “y_st2_{0}_{1}_{2}_n{3}c{4}_all_{5}_{6}.npy”.
  - Note that if the -aj (--AllJumps) option is used together with the “getY” mode, then relaxation vectors are computed for a state, as well as the all states that can be reached by a single vacancy jump out of that state (exit states).
    - These files have the names “y_st1_exits_{0}_{1}_{2}_n{3}c{4}_all_{5}_{6}.npy” and “y_st2_exits_{0}_{1}_{2}_n{3}c{4}_all_{5}_{6}.npy” for the exit states from the initial and final states of a single KMC step in our datasets respectively.
    - These exit state relaxations are used for the scaled residual bias method discussed in our main paper, and also illustrated with Jupyter notebooks in our data base for the Mn and vacancy diffusion.
  - Our database has the optimal computed relaxation vectors (that give the least validation set transport coefficient) for every system simulated in our paper, along with jupyter notebooks to show how they are computed by our neural networks and how they are used for the scaled residual bias correction method.

The next page onwards, all options in our training code are printed. The example job script (“job_scrip_NN.sb”) shows how to launch training (“train” mode), evaluation (“eval” mode) and relaxation vector computation (“getY” mode) runs.

```
usage: GCNetRun.py [-h] [-DP /path/to/data] [-prm] [-cr /path/to/crys/dat]
                   [-a0 float] [-m string] [-shf] [-rl int] [-rStart int]
```

```
                [-bt] [-nojsr] [-aos] [-nosym] [-l0] [-nl int] [-nch int]
                [-ncL int] [-scr] [-DPr] [-td int] [-tn int] [-sep int]
                [-eep int] [-sp string] [-vSp int] [-aj] [-ajn] [-nt int]
                [-i int] [-lr float] [-dcy float] [-bs float] [-wm float]
                [-ws float] [-d] [-dpf string]

Input parameters for using Gcnets for predicting relaxation vectors

optional arguments:
  -h, --help            show this help message and exit
  -DP /path/to/data, --DataPath /path/to/data
                        Path to Data file. (default: None)

  -prm, --Perm          Whether to mix up the data set if a permutation array
                        is found. (default: False)

  -cr /path/to/crys/dat, --CrysDatPath /path/to/crys/dat
                        Path to crystal Data. (default: None)

  -a0 float, --LatParam float
                        Lattice parameter. (default: 1.0)

  -m string, --Mode string
                        Running mode (one of train, eval, getY, getRep). If
                        getRep, then layer must specified with -RepLayer.
                        (default: None)

  -shf, --Shuffle       Whether to Shuffle at every epoch - applicable only to
                        training mode. (default: False)

  -rl int, --RepLayer int
                        Which Layer to extract representation from (count
                        starts from 0). Must be (0 or 2 or 5 or 8 ...)
                        (default: None)

  -rStart int, --RepStart int
                        Which sample to start computing representations onward
                        (N_train no. of samples will be computed). (default:
                        0)

  -bt, --BoundTrain
                        Whether to train using boundary state averages.
                        (default: False)

  -nojsr, --JumpSort
                        Whether to switch on/off sort jumps by rates. Not
                        doing it will cause symmetry to break. (default: True)

  -aos, --AddOnSitesJPINN
```

Whether to consider on sites along with vacancy sites
in JPINN. (default: False)

-nosym, --NoSymmetry
Whether to switch off all symmetry operations except
identity. (default: False)

-l0, --ScaleL0
Whether to scale transport coefficients during
training with uncorrelated value. (default: False)

-nl int, --Nlayers int
No. of intermediate layers of the neural network.
(default: 1)

-nch int, --Nchannels int
No. of representation channels in non-input layers.
(default: 4)

-ncL int, --NchLast int
No. channels of the last layers - how many vectors to
produce per site. (default: 1)

-scr, --Scratch Whether to create new network and start from scratch
(default: False)

-DPr, --DatPar Whether to use data parallelism. Note - does not work
for residual or subnet models. Used only in Train and
eval modes. (default: False)

-td int, --Tdata int Temperature (or composition for SR2) to read data from
(default: None)

-tn int, --TNet int Temperature (or composition for SR2) to use networks
from For example one can evaluate a network trained on
1073 K data, on the 1173 K data, for transfer training.
The directory in which the networks will be saved contains
this temperature in its name, while data sets are read with
the --Tdata option above.
(default: None)

-sep int, --Start_epoch int
Starting epoch (for training, this network will be
read in.) (default: None)

-eep int, --End_epoch int
Ending epoch (for training, this will be the last
epoch.) (default: None)

-sp string, --SpecTrain string
species to compute transport coefficients for.Order independent, string of integers(Eg, 1 for species 1 etc (default: None)

-vSp int, --VacSpec int
species index of vacancy, must match dataset, default 0 (default: 0)

-aj, --AllJumps
Whether to train on all jumps, or single selected jumps out of a state. (default: False)

-ajn, --AllJumpsNetType
Whether to use network trained on all jumps, or single selected jumps out of a state. (default: False)

-nt int, --N_train int
No. of training samples. (default: 10000)

-i int, --Interval int
Epoch intervals in which to save or load networks. (default: 1)

-lr float, --Learning_rate float
Learning rate for Adam algorithm. (default: 0.001)

-dcy float, --Decay float
Weight decay (L2 penalty for the weights). (default: 0.0005)

-bs float, --Batch_size float
size of a single batch of samples. (default: 128)

-wm float, --Mean_wt float
Initialization mean value of weights. (default: 0.02)

-ws float, --Std_wt float
Initialization standard dev of weights. (default: 0.2)

-d, --DumpArgs Whether to dump arguments in a file (default: False)

-dpf string, --DumpFile string
Name of file to dump arguments to (can be the jobID in a cluster for example). (default: None)