GNN for atom-wise force prediction

Lothar Bezzon

Introduction

The graph neural network (GNN) structure adapts perfectly to molecular dynamics simulations: interactions between atoms and their neighbors can be naturally represented as a graph, where each atom is a node and edge attributes contain information about interactions, such as distance or the presence of a specific bond.
Although this method has shown outstanding performances in simulation of quantum properties [1], it seems unable to produce the same speedups in classical molecular dynamics [2]. However, its application to this field is perfect for showing the flexibility of GNN.

Goal

I developed a GNN [3] (using torch and PyG) and trained it to predict forces acting in small samples of water molecules. Then I tested the model on a much bigger sample and on an argon sample. Following the success of GemNet [5], I also compared it with a smaller but locally rotational equivariant model but I didn't get better results (it has been shown, altough only in a specific case, that equivariance may not be as important as other structural choices [6]).

Data generation

All training and testing data are generated using LAMMPS [4]. Data and files to generate them are present in the github repository [3].

Training data

I simulated 216 water molecules at constant temperature and pressure (controlled by a Nosé-Hoover thermostat) with a time step of 2 ps. After a short period of equilibration, relevant data (atom id, molecule id, atom type, Cartesian coordinates, force Cartesian components) are saved for each atoms each 5 time steps. A total of 10100 frames are saved and are divided between train and validation data with a 9/1 ratio.

Test data

Test data consist in 100 frames of a 1600 water molecules simulation (to test scalability) and 100 frames of a liquid argon simulation (to test flexibility). Simulations details are as before except for argon which uses a 1 ps time step.

Data processing

As anticipated, nodes contain information about atoms type and edges about interactions. Input features are chosen to respect some symmetries of the system and to allow generalization to diverse inputs. For example, using atom positions in input would make it not translation invariant, as the output must be; also using just the modulus of the distance, without its direction, is wrong: the input would have rotation invariance while the output does not. Moreover, writing nodes as one-hot vectors for the atom type would in principle be right, but it would not allow an immediate generalization to different systems. Instead I chose to put in the nodes the parameters used to define long range interactions between atoms (charge for Coulomb potential, ϵ and σ for Lennard-Jones potential).
A similar approach should be followed in defining edges attributes but, since it is not necessary in the specific example I will show, the type of interaction is a one-hot vector which distinguishes i) atom not in the same molecule, ii) atom directly bonded (O-H in water), iii) 2^{nd} nearest neighbors in a molecule (two H in the same water molecule). Finally, this vector is concatenated with interatomic distance and its direction and only atoms nearer than a cutoff radius are connected by edges.

Model architecture

First model

The model consists in: inputs encoders, three message passing layers and a decoder (Figure 1).

- Encoders: they are two (one for nodes and one for edges features) multilayer perceptrons with three hidden layers; hidden and output dimension is 128 and between layers there are a PReLU activation layer and a 0.1 Droput.
- Message passing: nodes are updated by $\mathbf{x}'_i = \text{MLP}\left(\Sigma_{j\in\mathcal{N}(i)}\left\{\text{MLP}\left((\mathbf{x}_i * \mathbf{x}_j)\,||\,\mathbf{e}_{ji}\right)\right\}\right)$ where MLP is as before and $||$ represents concatenation. Second and third layers use residual connection and don't have bond type in $\mathbf{e}_{ji}$. Edges are not updated.
- Decoder: an MLP like the others which outputs the three predicted Cartesian components of the force on each atom.

Equivariant model

The main difference is in the first Message Passing layer: interatomic distance direction is not encoded together with the other edge attributes but is multiplied by the predicted force modulus during aggregation.

Training details

To train and evaluate the model, I used L1 distance between outputs and ground truths. Batch size is 16 and the optimizer is Adam with learning rate starting from 0.001 and decaying exponentially. A five epochs warm-up was scheduled, and during this stage an extra loss penalizing outputs far from zero was added. I scheduled 50 epochs but stopped after 36 upon reaching a plateau, and chose the model after 30 epochs (I saved a checkpoint every 6 epochs) (Figure 2).

Results

As expected, due to GNN architecture, the scalability of the model is very good: per-atom validation error is 4.18 kcal/mol/Å(on 216 water molecules frames), very close to the average error on 100 frames of 1600 water molecules simulations (4.24 kcal/mol/Å). On the other hand, the test on argon gives completely useless results. An explanation of this failing (but also an interesting analysis of the model) can be found looking at the individual interactions learned by the network (Figure 3). There we can see that the force fit is quite good in the range where the respective interaction can be found in the training set, except for the Lennard-Jones interaction (probably because it is weaker compared to the others and is "rare", being present only between oxygen atoms).

Fine-tuning

Finally, I tested if the model trained on water could be fine-tuned to address different molecules. In fact, even if the encoders and the first message passing layers are strongly dependent on the inputs, the decoder and the other message passing layers should remain quite similar. So I further trained the model only on a much smaller dataset of argon, obtaining a great improvement but still not an overall good result (Figure 4).

References

[1] https://doi.org/10.48550/arXiv.2102.03150

[2] https://doi.org/10.48550/arXiv.2112.03383

[3] https://github.com/LotharBezzon/GNNforMD

[4] https://doi.org/10.1016/j.cpc.2021.108171

[5] https://doi.org/10.48550/arXiv.2106.08903

[6] https://doi.org/10.48550/arXiv.2311.03094

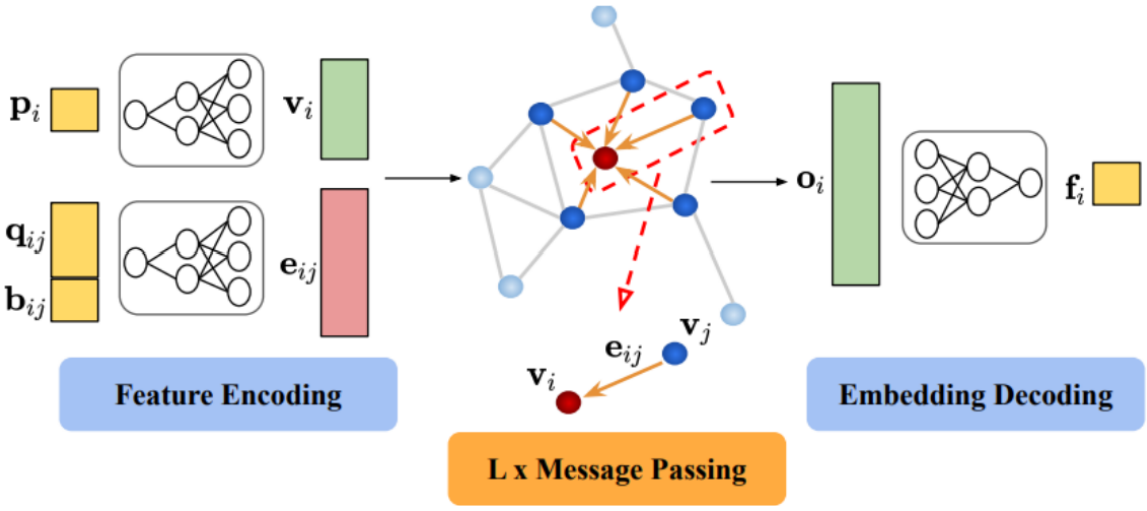

Figure 1: Schematic architecture of the network. Image from [2]

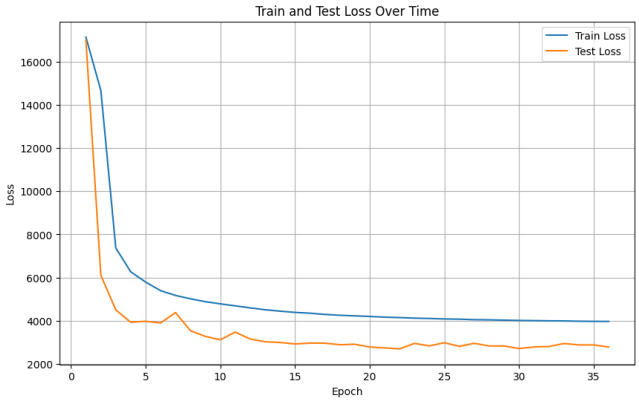

Figure 2: Train and validation losses over time. Validation error is always smaller due to the large dropout (10%).

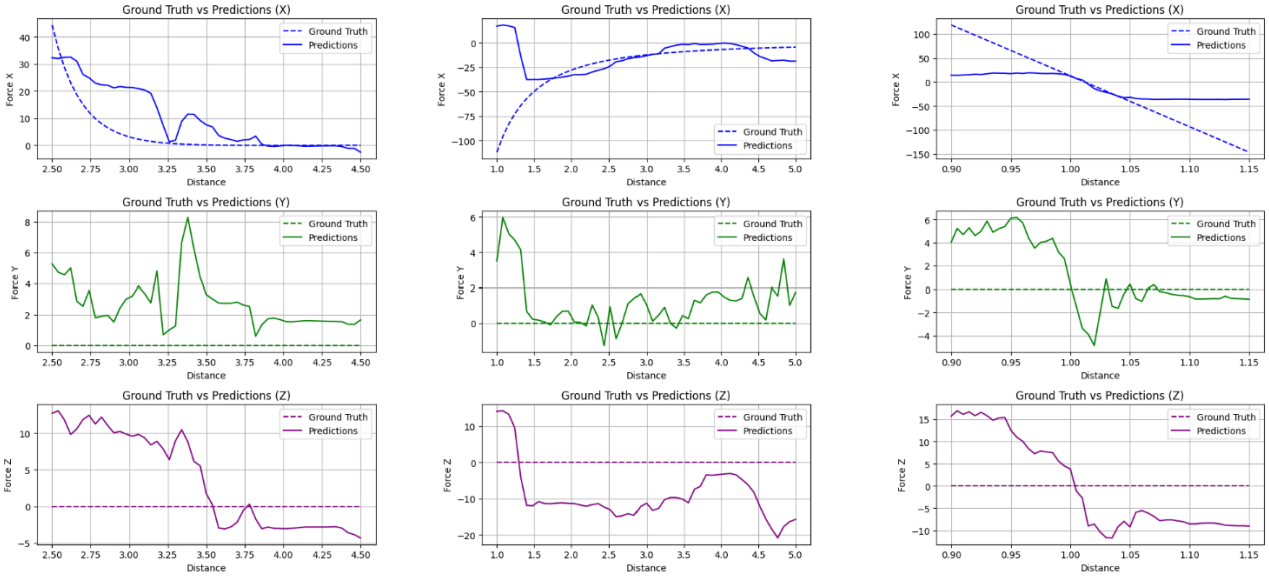

Figure 3: Testing how good did the model learn single interactions on two particles with similar characteristics to the ones in the training. From left to right: Lennard-Jones interaction, electrostatic interaction, harmonic covalent bond. Distances are measured in Åand forces in kcal/mol/Å.

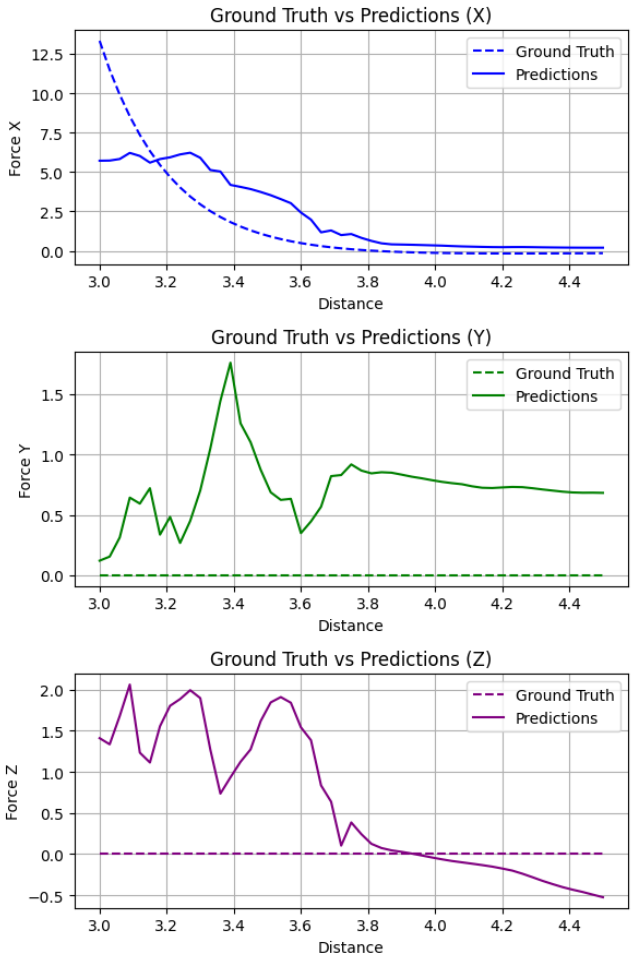

Figure 4: Learned Lennard-Jones interaction between argon atoms after a short fine-tuning of the model trained on water. Distances are measured in Åand forces in kcal/mol/Å.