nature computational science

Brief Communication https://doi.org/10.1038/s43588-023-00550-y

A deep learning model for predicting selected organic molecular spectra

In the format provided by the authors and unedited

Supplementary Information

Contents

Supplementary Section 1. Error Metrics.

The mean squared error (MSE) used as the loss function for training molecular properties is defined as:

$$L_y = \frac{1}{B}\sum_{b=1}^{B}\left(y_{out} - y_{ref}\right)^2$$

where y_{out} and y_{ref} represent the results of DetaNet's output and the reference values. B is the number of molecules in a training batch.

The MSE for atomic properties (such as charge) is defined by:

$$L_y = \frac{1}{I}\sum_{i=1}^{I}\left(y_{out} - y_{ref}\right)^2$$

Where I is the number of atoms in a training batch.

The coefficient of determination[1] (R^2) is used to assess accuracy:

$$R^2 = 1 - \sum_{b=1}^{B}\frac{\left(y_b^{out} - y_b^{ref}\right)^2}{\left(y_b^{out} - \frac{1}{B}\sum_{b=1}^{B} y_b^{out}\right)^2}$$

The Spearman coefficient[2] (Rs) indicates the accuracy of the DetaNet-predicted vibrational spectra compared to the DFT result; For two sets of data x_i and y_i, Rs is defined as:

$$Rs = 1 - \frac{6\sum_{i=1}^{n} d_i^2}{n(n^2 - 1)}$$

where d_i is the difference in magnitude between x_i and y_i, i is an index and n the total number of spectral data.

Supplementary Section 2. Vibration Spectra Prediction Details.

We used the chain rule to calculate infrared and Raman intensities. To be specific, the IR intensities and Raman activities are calculated as:

$$\frac{\partial \mu}{\partial \vec{P}_i} = C\left(\vec{P}_i\right) \times \frac{\partial \mu}{\partial \vec{r}_i}$$

$$\frac{\partial \alpha}{\partial \vec{P}_i} = C\left(\vec{P}_i\right) \times \frac{\partial \alpha}{\partial \vec{r}_i}$$

where $\vec{P}_i$ represents the normal coordinates that can be obtained by diagonalizing the Hessian matrix, C the Cartesian displacement, $\vec{r}_i$ stands for the Cartesian coordinate vector. After multiplying by a scaling factor of 0.965, the IR and Raman spectra are obtained by Lorentz broadening:

$$Lo(x) = \frac{F}{2\pi} \times \frac{1}{(x - x_n)^2 + 0.25 \times F} \times y_n$$

where F represents the half-width of the peak, which is set to 15 cm^{-1} for the infrared spectra and 10 cm^{-1} for the Raman spectra. x_n and y_n represents the calculated wavenumber and the IR/Raman intensity of the n^{th} vibrational mode, x is any wavenumber of the vibrational spectra. Both IR and Raman spectra have been normalized to facilitate comparison with experimental spectra.

Supplementary Section 3. UV-Vis Prediction Details.

As listed in **Supplementary Table 2**, we have tried many strategies to simulate the UV-Vis spectra. Firstly, we train the 10 excitation energies and the corresponding oscillator strengths at the same time. This strategy results an accuracy of R^2=0.95 and R^2=0.13 for the excitation energy and the oscillator strength. Then we adopted another strategy and separately train the excitation energies and the oscillator strength with the 10 values packed together. We found the accuracy is still low (R2=0.28) for the oscillator strength even though the excitation energy can be well predicted (R2=0.991). As listed in **Supplementary Table 2**, the one-by-one training on the 10 values of the oscillator strength can't increase the prediction accuracy. Instead of directly predicting the oscillator strength, we also trained the transition dipole vectors for the 10 excited states. Unfortunately, the transition dipole vectors can't be well reproduced (R^2=0.28), resulting the subsequently calculated oscillator strength can't be well predicted. Furthermore, we tried the phase-less loss function[3,4] developed by J Westermayr et al (J. Chem. Phys., 2020 153, 154112), but the accuracy is still low (R^2=0.41).

Through the above test, we found the transition dipole moment and oscillator strength can't be well learned. This can be attributed to the fact that the transition dipole moment depends on the wave functions of the excited and ground states. Similar conclusion[5] has been obtained by Ye et al. (PNAS, 2019, 116, 11612-11617), where they selected the Coulomb matrix that including more chemical information as the descriptors to predict the transition dipole moment. As a result, we tried directly train the Gaussian broaden UV-Vis spectra and found the prediction accuracy could reach 92% for oscillator strength. To be specific, the UV-Vis spectra are obtained as Gaussian broadening:

$$G(x) = \frac{y_n}{c\sqrt{2\pi}} e^{-\frac{(x-x_n)^2}{2c^2}} \quad c = \frac{F}{2\sqrt{ln_2}}$$

where F is the half-width of the Gaussian broadening and is set to 0.5 ev. x_n and y_n is the calculated wavelength and oscillator strength of the n^{th} excited states. x is any wavelength of the UV-Vis spectra. The spectra were broadened at 0.05 ev intervals in the range of 1.5 eV-13.5 eV; these adjusted data were used as the targets for training.

Supplementary Section 4. NMR Prediction Details.

It is well known that NMR spectra depend on the isotropic part of the shielding tensor. In this work, we compared two strategies to simulate NMR spectra. The first strategy is predicting the shielding tensors first and then calculating the isotropic parts. The second one is predicting the isotropic part of the shielding tensor directly. Taking carbon atoms as an example. we found that the second strategy (MAE=0.5332) performs better than the first one (MAE=2.4281). As a result, we first pre-trained a model to predict the isotropic part of the shielding tensor for all atoms. Then separate models were trained to predict the isotropic part of the shielding tensor for the four kinds of elements (H, C, N, O).

After predicting the isotropic part of the shielding tensors for carbon and hydrogen, we subtracted the reference values (C: 187.653, H: 31.751) from tetramethylsilane under the same calculation level (mpw1pw91/6-311+g(2d,p)) to obtain the NMR shifts. Finally, we summerized the ^{13}C and ^{1}H NMR shifts for the same chemical environment. The halfwidth used in the Lorentzian broadening was set to F=0.5 ppm for ^{13}C NMR and F=0.05 ppm for ^{1}H NMR.

In addition, we studied the performance of DetaNet in predicting NMR spectra for molecules in various solvent environments. As shown in **Supplementary Table 3**, the prediction error of the ^{13}C NMR spectra increases only about 0.00-0.03 ppm and ^{1}H NMR about 0.00-0.01 ppm compared to the gas phase, indicating the high accuracy and good transferability of DetaNet in describing the NMR spectra of molecules in solvents.

Supplementary Section 5. Transferability Test on DetaNet.

We assessed DetaNet's transferability by varying molecular types, sizes and geometries (**Supplementary Fig. 1a**). To do so, we randomly selected 5500 organic molecules (not included in QM9S) from the PubChem[6] database and performed geometrical optimization and spectra simulations. The average Spearman coefficients between DetaNet- and DFT-predicted IR and Raman spectra is 0.9879 and 0.9854, respectively, demonstrating that DetaNet achieves good transferability to molecules not included in the QM9S dataset. More importantly, DetaNet consistently produced precise spectra for a variety of molecular sizes. As shown in **Supplementary Fig. 1b**, the Spearman coefficients of infrared and Raman only decreased by 0.003 and 0.007 when the molecular size increased from 20 to 49 atoms. Here we illustruated the DetaNet and DFT predicted IR and Raman spectra for lactulose that contains 45 atoms (shown in **Supplementary Fig. 1c-d**). The almost identical results demonstrate DetaNet's transferability to larger molecules even though it was trained on QM9S, a dataset containing molecules with at most 20 atoms.

Then, we adopted both implicit and explicit models to reproduce molecular distortion and test DetaNet's transferability. For the implicit model, we considered five kinds of external environments including electric fields of 0.01 or 0.02 a.u., and the polarizable continuum effect with water, ethanol or dimethyl sulfoxide as the solvent. One of these five implicit environments was randomly applied to 5500 molecules; the molecular distortion degree $Dist$ so induced is calculated as

$$Dist = \frac{1}{2n(d)}\sum_{n(d)}\left|\frac{\Delta d}{d}\right| + \frac{1}{2n(a)}\sum_{n(a)}|sin(\Delta a)|$$

where d and a are the bond lengths and angles, respectively; Δd and Δa are the changes in bond lengths and angles induced by the external environment. Our findings indicate that a molecular distortion degree of 0.3 results in only a slight decrease of 0.011 and 0.004 in the accuracy of DetaNet for IR and Raman spectra, respectively (**Supplementary Fig. 1e**). Furthermore, the average accuracy of DetaNet for IR and Raman simulations was 98.30% and 98.38%, respectively, for all distorted molecules, demonstrating its strong transferability. The IR and Raman spectra of the 5500 organic molecules randomly selected from PubChem is available at figshare: https://doi.org/10.6084/m9.figshare.24235333.

For the explicit environments, we firstly performed geometrical optimization for ether associated with 10 water molecules or adsorbed on a gold surface. The geometrical optimization of the molecules adsorbed on surface were performed using the periodic boundary condition[7] (PBC) model implemented in VASP[8] package. Then we simulated the IR and Raman spectra of ether by removing all solvent water molecules and substrate. The IR and Raman intensities are calculated as:

$$I_{IR} \propto \left(\frac{\partial u_z}{\partial \vec{P}}\right)^2$$

$$I_{Raman} \propto \frac{(\omega-\omega_{\text{in}})^4}{\omega}\left(\frac{\partial \alpha_{ZZ}}{\partial \vec{P}}\right)^2 \frac{1}{1-exp\left(\frac{-hc\omega}{KT}\right)}$$

Here we only consider the u_z and α_{ZZ} for adsorption situations because the electric field is mainly along the z direction of the surface. As shown in **Supplementary Fig. 1f-g**, DFT and DetaNet reproduced the observed decreases in IR and Raman intensity of the methene C-H stretching mode (~2859 cm^{-1}) and the red shift of the C-O stretching mode (~1102 cm^{-1}) caused by solvent effect or surface adsorption compared to the gas phase. Even though a small discrepancy was found in the predicted frequency for surface adsorption, DetaNet still provided reasonably-precise peak positions and relative intensities for both IR and Raman spectra of ether in aqueous solution and adsorbtion on a gold surface.

To further verify the DetaNet's applicability to surface adsorption, more adsorbed molecules (formamide and urea) and substrates (gold and silver) are investigated. We can see from **Supplementary Fig. 2** that the DetaNet- and DFT-predicted IR and Raman spectra are almost identical, indicating the DetaNet is applicable to the various surface adsorption.

Supplementary Section 6. Molecular Properties Calculations Examples. Taking the water molecule as an example, here we illustrate how to calculate the total dipole moment. As shown in **Supplementary Fig. 3**, we firstly predicted the local and non-local dipole moment for every atom. Then the total dipole moment of the H_2O molecule can be obtained by summarizing the local and non-local dipole contributions.

It is noted that DetaNet is orientation sensitive because of it is rotational equivariant. This means that the geometrical rotation of the molecule will cause the rotation of equivariant properties such as the dipole moment and polarizability tensors. Here we have taken the water molecule for example to illustrate how the DetaNet is sensitive to orientation. As shown in **Supplementary Fig. 4**, rotating the H2O molecule will not change the norm of the dipole moment. However, the three components will change accordingly.

It is well known that the first hyperpolarizability is crucial for the development of optical data storage devices, and it determines the efficiency of a molecular switch in responding to an applied electric field or light. Here we taking enol-keto tautomer as example to test the accuracy of DetaNet in predicting the first hyperpolarizability. We found the DetaNet predicted first hyperpolarizability for 3-Methyl-2-butanone and 3-Methyl-2-buten-2-ol molecule is 380.9 and 481.5, compared to the value of 380.6 and 481.2 from DFT simulations. The small deviation of 0.03% and 0.04% between DetaNet and DFT simulations indicates the high preciseness of DetaNet and its broad application scenarios, as shown in **Supplementary Fig. 5.**

Supplementary Section 7. Effect of Cutoff Radius Values. The cutoff radius of 5.0 Å ensures the influence of the first-, second- and third-order neighboring atoms taken into account. Taking the NMR spectrum of the benzene molecule as an example, we did a contrast experiment by setting R_c=1.5, 2.5, 3.5 and 5.0 Å to consider the different-order neighbors. As shown in **Supplementary Fig. 6.**, the R_c of 1.5 Å only takes the 1st-order neighbors into account, while R_c=2.5 Å considers the 1st- and 2nd-order neighbors. The R_c=3.5 Å takes the 1st-, 2nd and 3rd-order neighbors into account, while R_c=5.0 Å may include 4th-order neighbors. We can see the predicted error of DetaNet compared to DFT is 10.6128, 2.2612, 0.0747 and 0.0747 ppm if the cut off value set to 1.5, 2.5, 3.5 and 5.0 Å. As a result, we can conclude that a cutoff radius of 5.0 Å is sufficient in the DetaNet model.

Supplementary Section 8. Comparison of DetaNet with Other MPNN Models. We trained and evaluated 4 advanced MPNNs model (on the QM9S dataset): Schnet[9], DimeNet++[10], PaiNN[11] and Nequip[12]. Four molecular properties including NPA charge, dipole moment, polarizability and first hyperpolarizability were considered. From the computational time listed in **Supplementary Table 5**, we can see that the efficiency of DetaNet is comparable or worse than the other methods. This can be attributed to the fact DetaNet passes not only scalar but also tensor messages. However, our model is much faster than both Nequip using tensors and DimeNet++ using angle features. This is because we simplified the tensor product and replaced the expensive feature cross product with linear layer operations. On the other hand, DetaNet achieves state-of-art performance in predicting the invariant scalars. Furthermore, DetaNet performs better than other MPNNs in reproducing the equivariant vectors or tensors.

Supplementary Section 9. IR/Raman Spectra Simulation of Flexible Molecules.
Taking the molecule 2-amino-3-(3,5-dioxo-1,2,4-oxadiazolidin-2-yl) propanoic acid as example, we tested the accuracy and transferability of the DetaNet in describing the flexible compounds. We firstly searched and optimized 6 different conformers for this molecule using Gaussian 16 package. Then both DFT and DetaNet are used to simulate the molecular properties and vibrational spectra for every conformer. As shown in **Supplementary Fig. 7**, we can see that the MAE of DetaNet in predicting the energy, dipole moment and polarizability is 0.029 ev, 0.16 Debye and 0.607 a.u. Furthermore, the DetaNet and DFT obtained almost identical IR and Raman spectra, indicating the DetaNet's ability to distinguish different conformers."

Supplementary Section 10. Possible Pitfalls of DetaNet. By simulating the IR and Raman spectra of the 5500 organic molecules extracted from the PubChem database and evaluation set of QM9S, we found two possible pitfalls of DetaNet. Firstly, DetaNet can't well reproduce the IR spectra for very small molecules with less than 5 atoms. As shown in **Supplementary Fig. 8 a-c**, the Spearman correlation between the DetaNet- and DFT-predicted IR spectra for ammonia, acetylene and formaldehyde molecule is 0.9813, 0.9605 and 0.9650, respectively. The underlying reason is that the Hessian matrix for very small molecules can't be well predicted. Secondly, As shown in **Supplementary Fig. 8 d-f**, DetaNet could not well reproduce the non-local vibrational modes of the six-membered cyclic aromatic hydrocarbons that usually located at around 700-800 cm^{-1}, that is, the rocking vibration of C-H perpendicular to the ring surface. This is cumulative effect because 10-12 atoms are involved in this vibration and the diagonalization of the Hessian matrix must bring big errors.

Supplementary Section 11. Details of the DetaNet code implementation.

11.1 Running environment.

Python 3.8.5 with python-libraries:

e3nn 0.4.4;

Matplotlib 3.3.4;

Numpy 1.20.1;

Torch 1.10.0;

torch-geometric 2.0.2;

torch_scatter 2.0.9;

torch_cluster 1.5.9;

torch_sparse 0.6.12;

torch_spline_conv 1.2.1.

11.2 Train the DetaNet model.

11.2.1 load the dataset

```
In [1]: import torch
        import torch.nn as nn
        import torch.nn.functional as F
        from detanet_model import *
        from torch_geometric.loader import DataLoader
```

```
In [2]: '''First, the pytorch library can be used to load the dataset, which consists of 130K molecules, after importing the pytorch library.'''
        dataset=torch.load('qm9s.pt')

        len(dataset)

Out[2]: 129817
```

where the cartesian coordinates, atomic numbers, edge indices and various properties of each molecule packaged in the geometric.data.Data file.

```
In [3]: dataset[0]

Out[3]: Data(edge_index=[2, 20], pos=[5, 3], number=1, smile='C', z=[5], quadrupole=[1, 3, 3], octapole=[1, 3, 3, 3], npacharge=[5], dipole=[1, 3],
        polar=[1, 3, 3], hyperpolar=[1, 3, 3, 3], energy=[1, 1], Hij=[20, 3, 3], Hi=[5, 3, 3], dedipole=[5, 3, 3], depolar=[5, 3, 6], tran_dipole=
        [1, 10, 3], tran_energy=[1, 10])
```

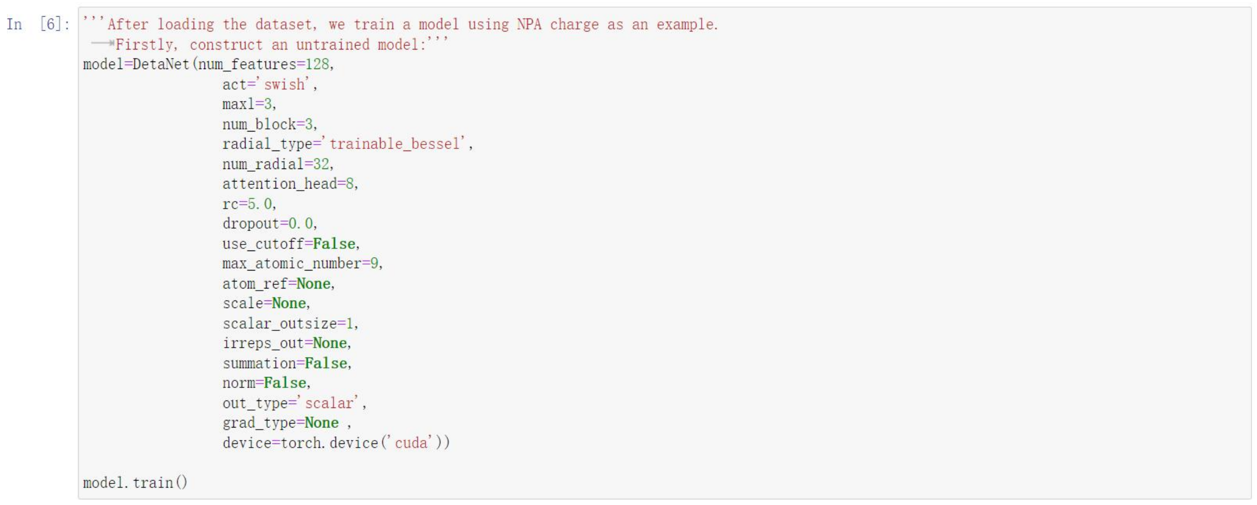
11.2.2 Divide the dataset into 5% testing and 95% training set:

```
In [4]: '''We divided the dataset evenly and used 5% of the data for testing and other for training:'''
        train_datasets=[]
        val_datasets=[]
        for i in range(len(dataset)):
            if i%20==0:
                val_datasets.append(dataset[i])
            else:
                train_datasets.append(dataset[i])

        len(train_datasets),len(val_datasets)

Out[4]: (123326, 6491)
```

11.2.3 Convert the dataset into a batch of 64 molecules of training:

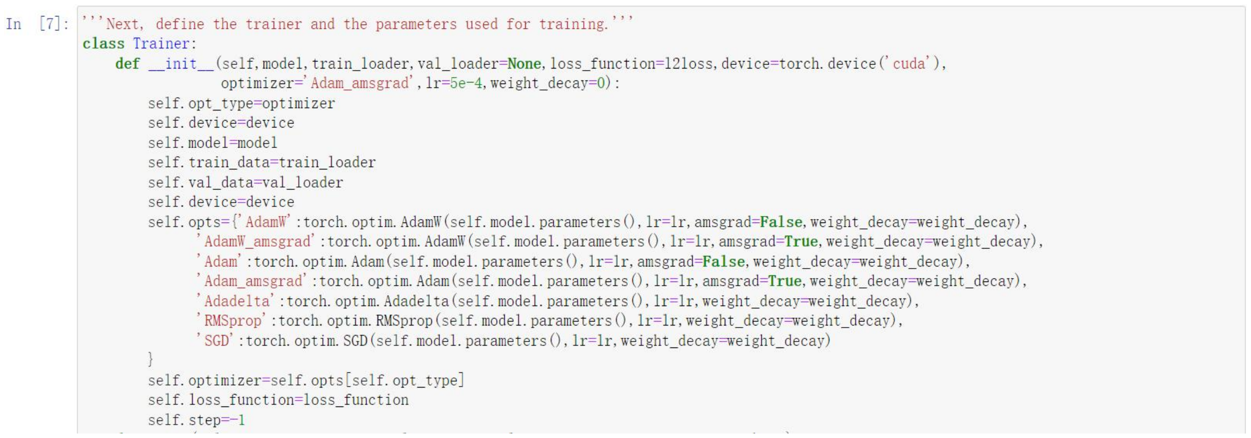
```
In [5]: '''Using torch_Geometric.dataloader.DataLoader Converts a dataset into a batch of 64 molecules of training data.'''
        bathes=64
        trainloader=DataLoader(train_datasets,batch_size=bathes,shuffle=True)
        valloader=DataLoader(val_datasets,batch_size=bathes,shuffle=True)
```

11.2.4 Construct an untrained model:

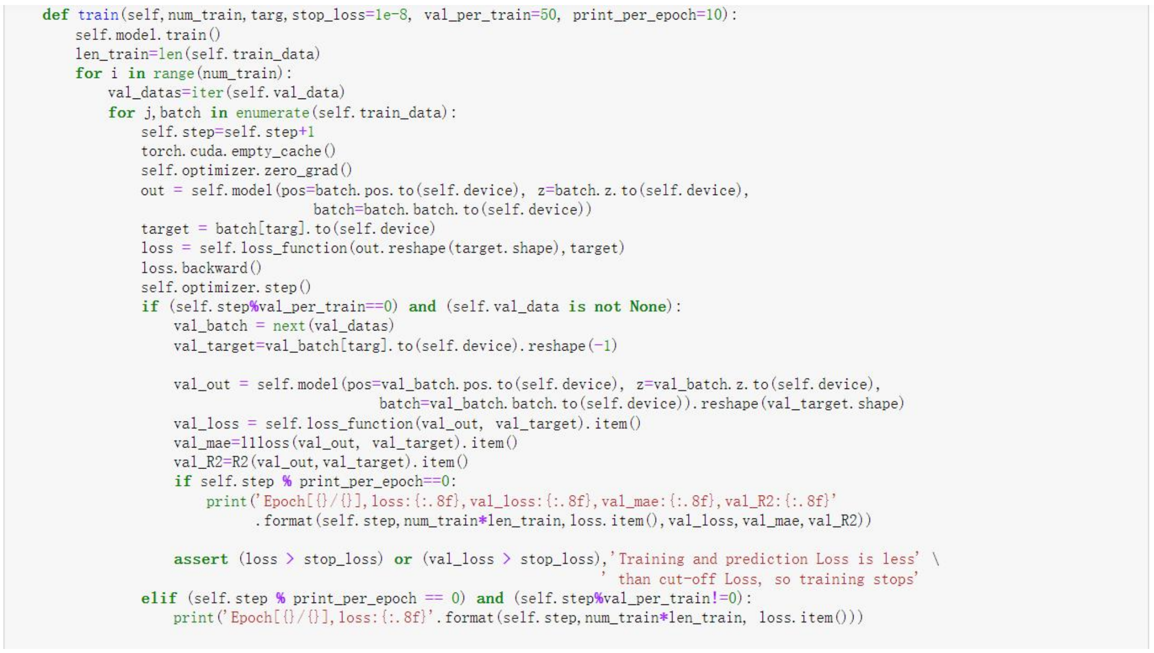


```
In  [6]: '''After loading the dataset, we train a model using NPA charge as an example.
          →Firstly, construct an untrained model:'''
         model=DetaNet(num_features=128,
                         act='swish',
                         maxl=3,
                         num_block=3,
                         radial_type='trainable_bessel',
                         num_radial=32,
                         attention_head=8,
                         rc=5.0,
                         dropout=0.0,
                         use_cutoff=False,
                         max_atomic_number=9,
                         atom_ref=None,
                         scale=None,
                         scalar_outsize=1,
                         irreps_out=None,
                         summation=False,
                         norm=False,
                         out_type='scalar',
                         grad_type=None ,
                         device=torch.device('cuda'))

         model.train()
```

11.2.5 Define the trainer and the parameters for training:



```
In  [7]: '''Next, define the trainer and the parameters used for training.'''
         class Trainer:
             def __init__(self,model,train_loader,val_loader=None,loss_function=l2loss,device=torch.device('cuda'),
                          optimizer='Adam_amsgrad',lr=5e-4,weight_decay=0):
                 self.opt_type=optimizer
                 self.device=device
                 self.model=model
                 self.train_data=train_loader
                 self.val_data=val_loader
                 self.device=device
                 self.opts={'AdamW':torch.optim.AdamW(self.model.parameters(),lr=lr,amsgrad=False,weight_decay=weight_decay),
                       'AdamW_amsgrad':torch.optim.AdamW(self.model.parameters(),lr=lr,amsgrad=True,weight_decay=weight_decay),
                       'Adam':torch.optim.Adam(self.model.parameters(),lr=lr,amsgrad=False,weight_decay=weight_decay),
                       'Adam_amsgrad':torch.optim.Adam(self.model.parameters(),lr=lr,amsgrad=True,weight_decay=weight_decay),
                       'Adadelta':torch.optim.Adadelta(self.model.parameters(),lr=lr,weight_decay=weight_decay),
                       'RMSprop':torch.optim.RMSprop(self.model.parameters(),lr=lr,weight_decay=weight_decay),
                       'SGD':torch.optim.SGD(self.model.parameters(),lr=lr,weight_decay=weight_decay)
                 }
                 self.optimizer=self.opts[self.opt_type]
                 self.loss_function=loss_function
                 self.step=-1
```



```
    def train(self,num_train,targ,stop_loss=1e-8, val_per_train=50, print_per_epoch=10):
        self.model.train()
        len_train=len(self.train_data)
        for i in range(num_train):
            val_datas=iter(self.val_data)
            for j,batch in enumerate(self.train_data):
                self.step=self.step+1
                torch.cuda.empty_cache()
                self.optimizer.zero_grad()
                out = self.model(pos=batch.pos.to(self.device), z=batch.z.to(self.device),
                                 batch=batch.batch.to(self.device))
                target = batch[targ].to(self.device)
                loss = self.loss_function(out.reshape(target.shape),target)
                loss.backward()
                self.optimizer.step()
                if (self.step%val_per_train==0) and (self.val_data is not None):
                    val_batch = next(val_datas)
                    val_target=val_batch[targ].to(self.device).reshape(-1)

                    val_out = self.model(pos=val_batch.pos.to(self.device), z=val_batch.z.to(self.device),
                                         batch=val_batch.batch.to(self.device)).reshape(val_target.shape)
                    val_loss = self.loss_function(val_out, val_target).item()
                    val_mae=l1loss(val_out, val_target).item()
                    val_R2=R2(val_out,val_target).item()
                    if self.step % print_per_epoch==0:
                        print('Epoch[{}/{}],loss:{:.8f},val_loss:{:.8f},val_mae:{:.8f},val_R2:{:.8f}'
                              .format(self.step,num_train*len_train,loss.item(),val_loss,val_mae,val_R2))

                    assert (loss > stop_loss) or (val_loss > stop_loss),'Training and prediction Loss is less' \
                                                                         ' than cut-off Loss, so training stops'
                elif (self.step % print_per_epoch == 0) and (self.step%val_per_train!=0):
                    print('Epoch[{}/{}],loss:{:.8f}'.format(self.step,num_train*len_train, loss.item()))
```

11.2.6 Modify the data type and device type:

```
In [8]: '''Then, modify the data type and device type'''
        device=torch.device('cuda')
        dtype=torch.float32
        model=model.to(dtype)
        model=model.to(device)
```

11.2.7 Train 20 times from a 5e-4 learning rate and AdamW optimizer:

```
In [9]: '''Finally, using the trainer, training 20 times from a 5e-4 learning rate'''
        trainer=Trainer(model,train_loader=trainloader,val_loader=valloader,loss_function=l2loss,lr=5e-4,weight_decay=0,optimizer='AdamW')
```

```
In [10]: trainer.train(num_train=20,targ='npacharge')
```

```
Epoch[0/1927],loss:0.06453520,val_loss:0.09647646,val_mae:0.26150602,val_R2:0.05411941
Epoch[10/1927],loss:0.03383980
Epoch[20/1927],loss:0.02285156
Epoch[30/1927],loss:0.01284886
Epoch[40/1927],loss:0.01130290
Epoch[50/1927],loss:0.00883452,val_loss:0.01035036,val_mae:0.06626014,val_R2:0.90387326
Epoch[60/1927],loss:0.00643805
Epoch[70/1927],loss:0.00432105
Epoch[80/1927],loss:0.00278638
Epoch[90/1927],loss:0.00219296
Epoch[100/1927],loss:0.00170563,val_loss:0.00196205,val_mae:0.02748346,val_R2:0.98102534
Epoch[110/1927],loss:0.00193614
Epoch[120/1927],loss:0.00124189
Epoch[130/1927],loss:0.00122081
Epoch[140/1927],loss:0.00109711
Epoch[150/1927],loss:0.00093220,val_loss:0.00098367,val_mae:0.02152115,val_R2:0.99056077
Epoch[160/1927],loss:0.00097455
Epoch[170/1927],loss:0.00074644
Epoch[180/1927],loss:0.00097347
```

11.2.8 Save and load the trained parameters:

```
In [11]: '''After the training is completed, take out the learnable parameters and save them as a .pth file.'''
         state_dict=model.state_dict

         state_dict()
```

```
Out[11]: OrderedDict([('Embedding.act.alpha',
                       tensor([0.9962, 0.9973, 0.9957, 0.9963, 0.9970, 0.9955, 0.9992, 0.9982, 0.9989,
                               0.9986, 0.9998, 0.9981, 0.9985, 1.0003, 0.9984, 0.9977, 0.9996, 0.9986,
                               0.9977, 0.9981, 0.9999, 0.9982, 1.0006, 1.0010, 0.9995, 0.9943, 0.9993,
                               0.9981, 1.0002, 0.9980, 0.9988, 0.9987, 0.9986, 0.9991, 0.9979, 1.0001,
                               0.9977, 0.9989, 0.9933, 1.0012, 0.9985, 0.9986, 1.0006, 1.0005, 0.9991,
                               0.9958, 0.9984, 0.9982, 0.9971, 0.9978, 0.9963, 0.9995, 0.9972, 0.9947,
                               0.9982, 0.9947, 0.9979, 1.0004, 0.9943, 0.9986, 0.9994, 0.9996, 0.9954,
                               1.0031, 0.9999, 0.9978, 0.9997, 0.9979, 1.0004, 0.9998, 0.9987, 0.9994,
                               0.9980, 0.9966, 0.9999, 0.9983, 0.9955, 0.9986, 0.9960, 0.9936, 1.0008,
                               0.9938, 0.9968, 0.9962, 0.9958, 0.9984, 0.9957, 1.0001, 0.9967, 0.9959,
                               0.9994, 0.9981, 0.9956, 0.9963, 0.9987, 0.9982, 0.9934, 0.9995, 0.9982,
                               0.9979, 0.9978, 0.9980, 0.9993, 0.9997, 0.9994, 0.9987, 0.9982, 0.9981,
                               0.9946, 0.9959, 0.9992, 0.9975, 0.9977, 0.9971, 1.0007, 0.9945, 0.9958,
                               1.0002, 1.0001, 0.9965, 0.9989, 0.9922, 0.9941, 0.9975, 0.9988, 1.0004,
                               0.9989, 0.9939], device='cuda:0')),
                      ('Embedding.act.beta',
                       tensor([1.7003, 1.7009, 1.6966, 1.6985, 1.6990, 1.7053, 1.7019, 1.7040, 1.7023,
                               1.7014, 1.7005, 1.7022, 1.7007, 1.7020, 1.7014, 1.7017, 1.7023, 1.7007,
```

```
In [12]: torch.save(model.state_dict(),'npacharge_param.pth')
```

```
In [13]: '''When needed model , simply load parameters on the untrained model. The parameters we have trained are saved in trained_Param/.'''
         state_dict=torch.load('npacharge_param.pth')
         model.load_state_dict(state_dict)

Out[13]: <All keys matched successfully>
```

11.3 Simulate the molecular properties and spectra using trained DetaNet.

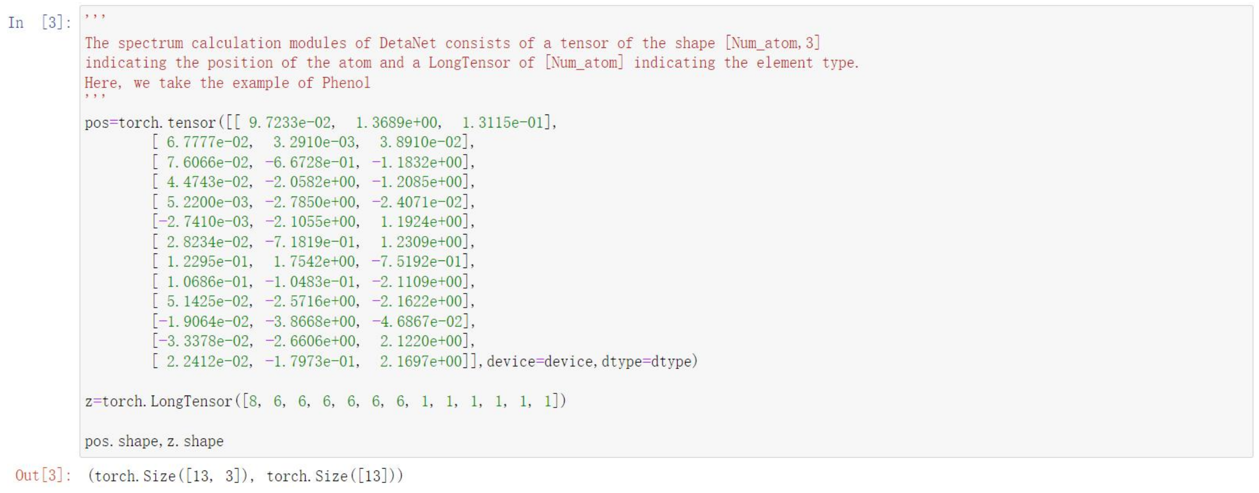
11.3.1 Import relevant libraries from the trained DetaNet model:

```
In [1]: import torch
        from detanet_model import nn_vib_analysis,Lorenz_broadening,DetaNet,get_raman_intensity,uv_model,nmr_calculator,nmr_sca,charge_model
```

11.3.2 Formulate the device and data type:

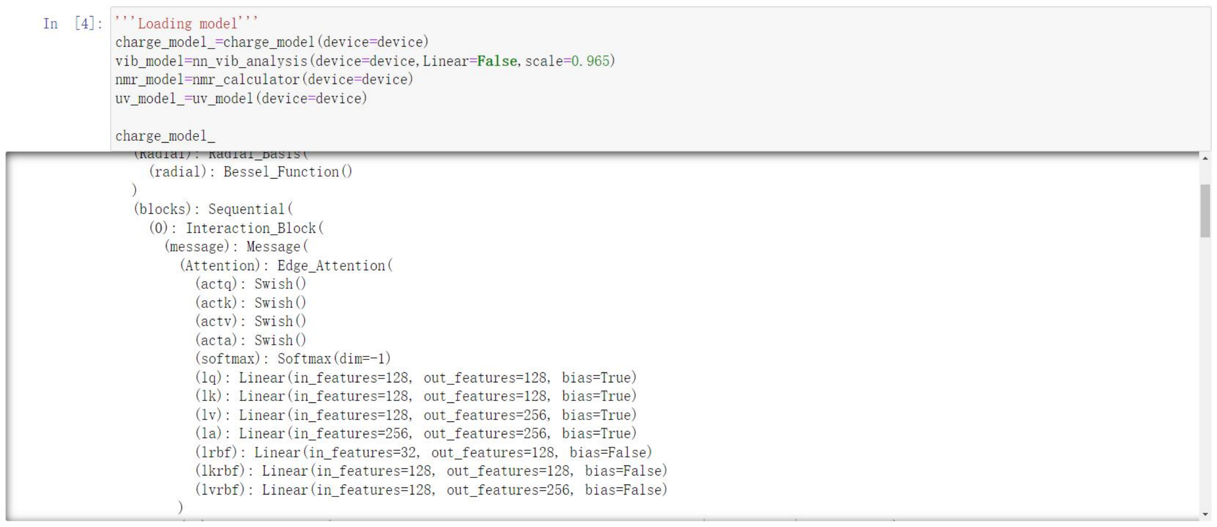
```
In [2]: device=torch.device('cpu')
        dtype=torch.float32
```

11.3.3 Import the cartesian coordinates:



```
In  [3]: '''
         The spectrum calculation modules of DetaNet consists of a tensor of the shape [Num_atom,3]
         indicating the position of the atom and a LongTensor of [Num_atom] indicating the element type.
         Here, we take the example of Phenol
         '''
         pos=torch.tensor([[ 9.7233e-02,  1.3689e+00,  1.3115e-01],
                 [ 6.7777e-02,  3.2910e-03,  3.8910e-02],
                 [ 7.6066e-02, -6.6728e-01, -1.1832e+00],
                 [ 4.4743e-02, -2.0582e+00, -1.2085e+00],
                 [ 5.2200e-03, -2.7850e+00, -2.4071e-02],
                 [-2.7410e-03, -2.1055e+00,  1.1924e+00],
                 [ 2.8234e-02, -7.1819e-01,  1.2309e+00],
                 [ 1.2295e-01,  1.7542e+00, -7.5192e-01],
                 [ 1.0686e-01, -1.0483e-01, -2.1109e+00],
                 [ 5.1425e-02, -2.5716e+00, -2.1622e+00],
                 [-1.9064e-02, -3.8668e+00, -4.6867e-02],
                 [-3.3378e-02, -2.6606e+00,  2.1220e+00],
                 [ 2.2412e-02, -1.7973e-01,  2.1697e+00]],device=device,dtype=dtype)

         z=torch.LongTensor([8, 6, 6, 6, 6, 6, 6, 1, 1, 1, 1, 1, 1])

         pos.shape,z.shape

Out[3]:  (torch.Size([13, 3]), torch.Size([13]))
```

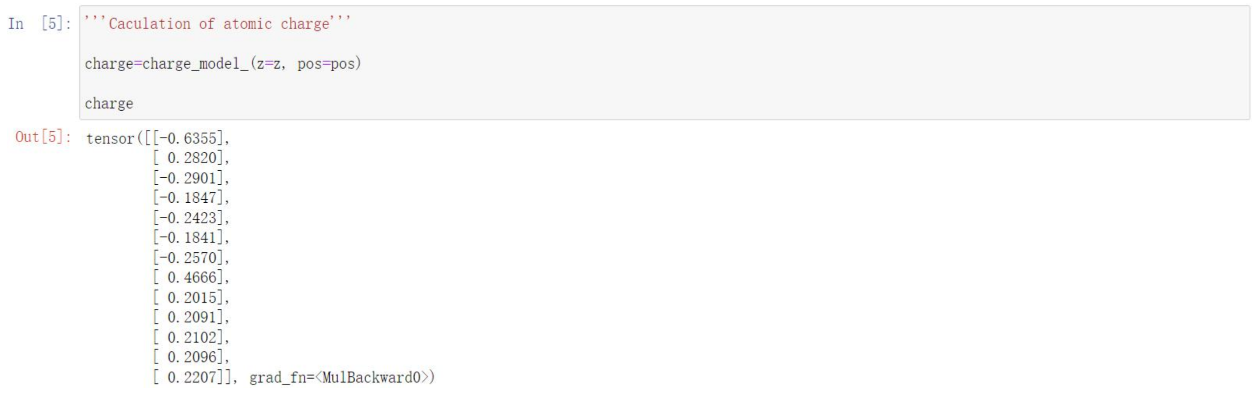
11.3.4 Load the trained DetaNet models:



```
In  [4]: '''Loading model'''
         charge_model_=charge_model(device=device)
         vib_model=nn_vib_analysis(device=device,Linear=False,scale=0.965)
         nmr_model=nmr_calculator(device=device)
         uv_model_=uv_model(device=device)

         charge_model_
```

```
      (Radial): Radial_Basis(
        (radial): Bessel_Function()
      )
      (blocks): Sequential(
        (0): Interaction_Block(
          (message): Message(
            (Attention): Edge_Attention(
              (actq): Swish()
              (actk): Swish()
              (actv): Swish()
              (acta): Swish()
              (softmax): Softmax(dim=-1)
              (lq): Linear(in_features=128, out_features=128, bias=True)
              (lk): Linear(in_features=128, out_features=128, bias=True)
              (lv): Linear(in_features=128, out_features=256, bias=True)
              (la): Linear(in_features=256, out_features=256, bias=True)
              (lrbf): Linear(in_features=32, out_features=128, bias=False)
              (lkrbf): Linear(in_features=128, out_features=128, bias=False)
              (lvrbf): Linear(in_features=128, out_features=256, bias=False)
            )
```

11.3.5 Calculate the NPA charge:



```
In  [5]: '''Caculation of atomic charge'''

         charge=charge_model_(z=z, pos=pos)

         charge

Out[5]:  tensor([[-0.6355],
                 [ 0.2820],
                 [-0.2901],
                 [-0.1847],
                 [-0.2423],
                 [-0.1841],
                 [-0.2570],
                 [ 0.4666],
                 [ 0.2015],
                 [ 0.2091],
                 [ 0.2102],
                 [ 0.2096],
                 [ 0.2207]], grad_fn=<MulBackward0>)
```

11.3.6 Calculate the IR intensities and Raman activities:

```
In  [6]: '''Calculation of frequency, IR intensity, Raman activity for each vibration mode. (Units are consistent with Gaussian G16)'''
         freq,iir,araman=vib_model(z=z, pos=pos)
         for i in range(len(freq)):
             print('Vibration mode:{},Frequency:{:.8f},IR intensity:{:.8f},Raman activity:{:.8f}'
               .format(i+1,freq[i].item(),iir[i].item(),araman[i].item()))
```

```
Vibration mode:1,Frequency:234.93307495,IR intensity:58.49052811,Raman activity:0.66419905
Vibration mode:2,Frequency:240.97242737,IR intensity:54.48851395,Raman activity:3.02271128
Vibration mode:3,Frequency:389.43359375,IR intensity:0.03612965,Raman activity:0.00669313
Vibration mode:4,Frequency:404.78579712,IR intensity:11.21769810,Raman activity:0.37456882
Vibration mode:5,Frequency:489.18301392,IR intensity:1.03978169,Raman activity:0.16590688
Vibration mode:6,Frequency:521.72906494,IR intensity:2.17546701,Raman activity:5.68891144
Vibration mode:7,Frequency:613.23931885,IR intensity:6.04010534,Raman activity:0.05951004
Vibration mode:8,Frequency:615.38885498,IR intensity:0.39308357,Raman activity:5.37667418
Vibration mode:9,Frequency:758.73803711,IR intensity:92.25366974,Raman activity:0.01445770
Vibration mode:10,Frequency:784.02593994,IR intensity:9.00050354,Raman activity:0.10011159
Vibration mode:11,Frequency:803.38812256,IR intensity:21.92415237,Raman activity:18.16299438
Vibration mode:12,Frequency:811.47283936,IR intensity:0.16154622,Raman activity:0.08204326
Vibration mode:13,Frequency:851.58459473,IR intensity:0.14443359,Raman activity:0.02477967
Vibration mode:14,Frequency:862.03979492,IR intensity:0.37681001,Raman activity:0.01351147
Vibration mode:15,Frequency:985.88153076,IR intensity:3.15740442,Raman activity:30.39320564
Vibration mode:16,Frequency:1009.12152100,IR intensity:2.94540358,Raman activity:10.95648479
Vibration mode:17,Frequency:1046.92773438,IR intensity:21.50440025,Raman activity:0.79917258
Vibration mode:18,Frequency:1140.75756836,IR intensity:35.87324905,Raman activity:4.98619318
Vibration mode:19,Frequency:1146.94750977,IR intensity:123.70233154,Raman activity:3.44818425
Vibration mode:20,Frequency:1154.11315918,IR intensity:1.57686353,Raman activity:2.45739865
Vibration mode:21,Frequency:1233.63439941,IR intensity:74.17893219,Raman activity:10.94951725
Vibration mode:22,Frequency:1295.32153320,IR intensity:12.15853024,Raman activity:0.91542810
Vibration mode:23,Frequency:1332.42919922,IR intensity:21.27632141,Raman activity:0.37781754
Vibration mode:24,Frequency:1451.45507812,IR intensity:24.00309753,Raman activity:0.32206699
Vibration mode:25,Frequency:1477.11767578,IR intensity:56.28190613,Raman activity:1.43802309
Vibration mode:26,Frequency:1579.03259277,IR intensity:27.14629364,Raman activity:10.53184986
Vibration mode:27,Frequency:1595.42626953,IR intensity:47.80065536,Raman activity:15.81441593
Vibration mode:28,Frequency:3043.97241211,IR intensity:15.00514793,Raman activity:74.13775635
Vibration mode:29,Frequency:3066.66381836,IR intensity:0.18411070,Raman activity:73.36512756
Vibration mode:30,Frequency:3072.03344727,IR intensity:13.47472286,Raman activity:75.83457947
Vibration mode:31,Frequency:3082.52807617,IR intensity:13.22551155,Raman activity:52.49898911
Vibration mode:32,Frequency:3090.39160156,IR intensity:2.94092011,Raman activity:274.87701416
Vibration mode:33,Frequency:3679.24731445,IR intensity:53.40877914,Raman activity:109.73700714
```

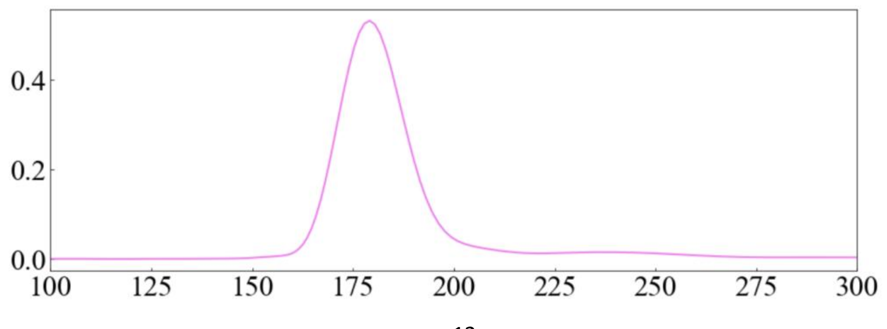
11.3.7 Simulate the NMR Shift:

```
In  [12]: '''Aggregation of hydrogen and carbon from the same chemical environment
          in index,all hydrogen atoms and carbon atoms in the same chemical environment are represented by same number

          '''
          indexc=torch.LongTensor([0,1,2,3,2,1])
          indexh=torch.LongTensor([0,1,2,3,2,1])
          shiftc,intc,shifth,inth=nmr_sca(sc.reshape(-1),sh.reshape(-1),indexc,indexh)
```

```
In  [13]: '''Broadening of nmr shift and intensity'''
          xh=torch.linspace(0, 12, 1201)
          xc=torch.linspace(0,300,3001)
          yc=Lorenz_broadening(shiftc,intc,c=xc,sigma=1).detach().numpy()
          yh=Lorenz_broadening(shifth,inth,c=xh,sigma=0.05).detach().numpy()
```

11.3.8 Simulate the UV-Vis spectra:

```
In  [17]: '''draw uv spectrum'''
          xuv=1239.85/torch.linspace(1.5,13.5,240).detach().numpy()
          plt.figure(figsize=(15,5))
          plt.rc('font',family='Times New Roman', size=30)
          plt.xlim(100,300)
          plt.plot(xuv,uv,lw=2,color='Violet')

Out[17]: [<matplotlib.lines.Line2D at 0x11e5efff940>]
```



Supplementary Tables

Supplementary Table 1. Different molecular properties predicted by DetaNet. The QM9S, QM9-NMR, QM9 and QM7-X dataset are used as the reference data to enable DetaNet to predict the spectra-related, NMR-related, energy-related, and MD-related molecular properties. Here MAE, RMSE and R^2 represent the mean absolute errors, the root mean square errors and the coefficients of determination, respectively.

Dataset	Property	Symbol	Unit	MAE	RMSE	R^2
QM9S (130K data)	Natural population charge	q_i^{NPA}	e	0.0011	0.0017	0.99998
	Dipole Moment	$\vec{\mu}$	D	0.0088	0.1280	0.99994
	Polarizability Tensor	α	${a_0}^3$	0.1374	0.2227	0.99996
	First Hyperpolarizability	β	$a.u.$	2.0373	2.8950	0.99398
	Quadrupole Moment	Q	$D \cdot Å$	0.3834	0.6258	0.99945
	Octupole Moment	O	$D \cdot Å^2$	6.2442	13.116	0.97030
	Derivatives of Dipole	$\partial \mu / \partial \vec{r}_i$	$D/Å$	0.0181	0.0268	0.99943
	Derivatives of Polarizablity	$\partial \alpha / \partial \vec{r}_i$	${a_0}^3/Å$	0.2012	0.3087	0.99710
	Hessian (atomic)	H_i	$ev/Å^2$	0.0731	0.1341	0.99937
	Hessian (inter-atomic)	H_{ij}	$ev/Å^2$	0.0699	0.1145	0.99944
	Transition Energy	ΔE	ev	0.0768	0.1043	0.99138
	UV-Vis (1.5-13.5ev)	uv	$a.u.$	0.0025	0.0073	0.92026
QM9-NMR (130K data)	Shielding Isotropic All	σ^{iso}	ppm	0.4938	1.1322	0.99983
	Shielding Isotropic (H)	σ_H^{iso}	ppm	0.0543	0.0778	0.99846
	Shielding Isotropic (O)	σ_O^{iso}	ppm	1.6982	2.7342	0.99983
	Shielding Isotropic (N)	σ_N^{iso}	ppm	0.9866	1.7183	0.99979
	Shielding Isotropic (C)	σ_C^{iso}	ppm	0.5203	0.8822	0.99971
QM9 (134K data)	Homo energy	ϵ_{homo}	ev	0.0278	0.0452	0.99643
	Lumo energy	ϵ_{lumo}	ev	0.0169	0.0251	0.99963
	Gap energy	ϵ_{gap}	ev	0.0323	0.0501	0.99902
	Zero point vibration energy	$zpve$	ev	0.0012	0.0019	0.99999
	Free energy	H	ev	0.0076	0.0124	0.99999
	Enthalpy	G	ev	0.0087	0.0141	0.99999
	Electronic spatial extent	r^2	${a_0}^2$	0.0680	0.1217	0.99999
	Heat capacity	C_v	$cal/mol \cdot k$	0.0021	0.0035	0.99998
Partial QM7-X (144K data)	Energy	E	ev	0.0293	0.04019	0.99998
	Atomic Force	f_i	$ev/Å$	0.0341	0.05164	0.99905
	Hirshfeld charges	q_i^H	e	0.0011	0.0017	0.99998
	Atomic C6 Coefficient	C_6	$E_h \cdot {a_0}^6$	5.6112	8.3795	0.99984
	Moment of Inertia Tensor	I	$amu \cdot Å^2$	0.0958	0.1630	0.99999

Supplementary Table 2. Prediction accuracy for excitation energy and oscillator strength using different strategies. All the methods that separately predict the excitation energies and the corresponding oscillator strengths (or transition dipole vectors) falls in reproducing the UV-Vis spectra. Alternatively, we tried directly train the Gaussian broaden UV-Vis spectra and found the prediction accuracy could reach 92% for oscillator strength. Here the R^2 represent the coefficients of determination.

Prediction methods	Prediction R^2 for the Excitation energy	Prediction R^2 for the oscillator strength
Simultaneous training on the 10 excitation energies and the oscillator strengths	0.956	0.134
Separate training on the excitation energies and the oscillator strength (10 values packed together)	0.991	0.276
Separate training on the excitation energies and the transition dipole vectors (10 values packed together)	0.991	0.279
Separate training on the excitation energies and the oscillator strength (10 values one by one)	0.992	0.284
Using the phase-less loss function to predict dipole developed by J Westermayr et al	0.991	0.417
Direct training on the Gaussian broaden UV-Vis spectra	0.920	

Supplementary Table 3. Comparison of the prediction errors for NMR shifts using DetaNet for molecules in 6 different environments. The tetrachloromethane (CCl4), tetrahydrofuran (THF), toluene, ethanol and dimethyl sulfoxide (DMSO) with different dielectric constants (ϵ) were selected as the solvent. The prediction error (MAE) for the ^{13}C NMR and ^{1}H NMR spectra in various solvent environments increases only about 0.00-0.03 and 0.00-0.01 ppm compared to the gas phase.

Solvent	Gas	CCl_4	THF	Toluene	Ethanol	DMSO
ϵ	0	2.228	7.426	20.493	32.613	46.826
MAE(NMR^{1}H)	0.0543	0.0545	0.0551	0.0550	0.0547	0.0557
MAE(NMR^{13}C)	0.5203	0.5339	0.5452	0.5298	0.5396	0.5547

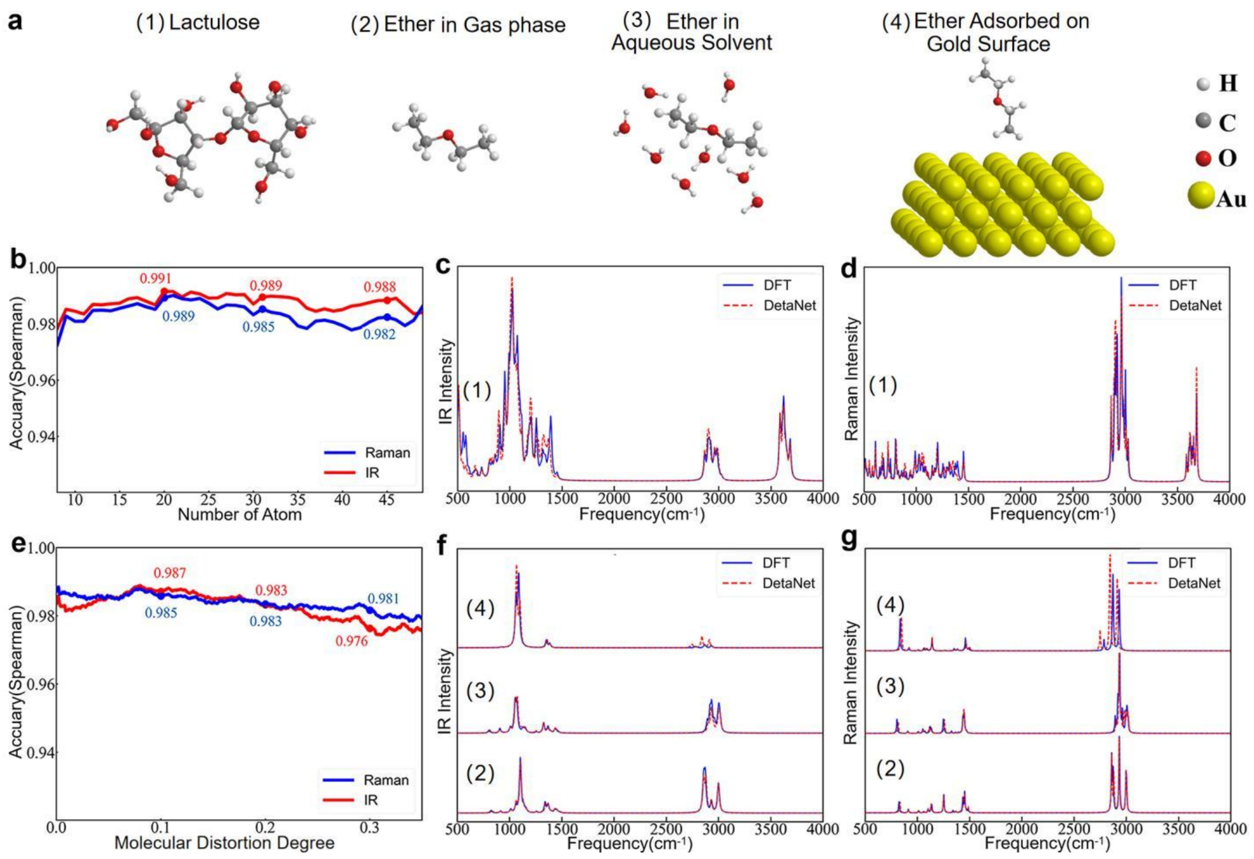
Supplementary Table 4. Electronic features of the seven common elements. Here we used the number of the paired and the single electrons of every electron subshell (1s, 2s, 2p, 3s, 3p) to construct the electronic features for element H, C, N, O, F, S and Cl. These features provide the detailed information of every subshell, as well as the spin and charge information. The electronic features were normalized and fed into the linear layer. Here Q_{pa} and Q_{si} represent the number of the paired and the unpaired electrons on every electronic subshell.

Element	**1s**		**2s**		**2p**		**3s**		**3p**	
	Q_{pa}^{1s}	Q_{si}^{1s}	Q_{pa}^{2s}	Q_{si}^{2s}	Q_{pa}^{2p}	Q_{si}^{2p}	Q_{pa}^{3s}	Q_{si}^{3s}	Q_{pa}^{3p}	Q_{si}^{3p}
H	0	1	0	0	0	0	0	0	0	0
C	2	0	2	0	0	2	0	0	0	0
N	2	0	2	0	0	3	0	0	0	0
O	2	0	2	0	2	2	0	0	0	0
F	2	0	2	0	4	1	0	0	0	0
S	2	0	2	0	6	0	2	0	2	2
Cl	2	0	2	0	6	0	2	0	4	1

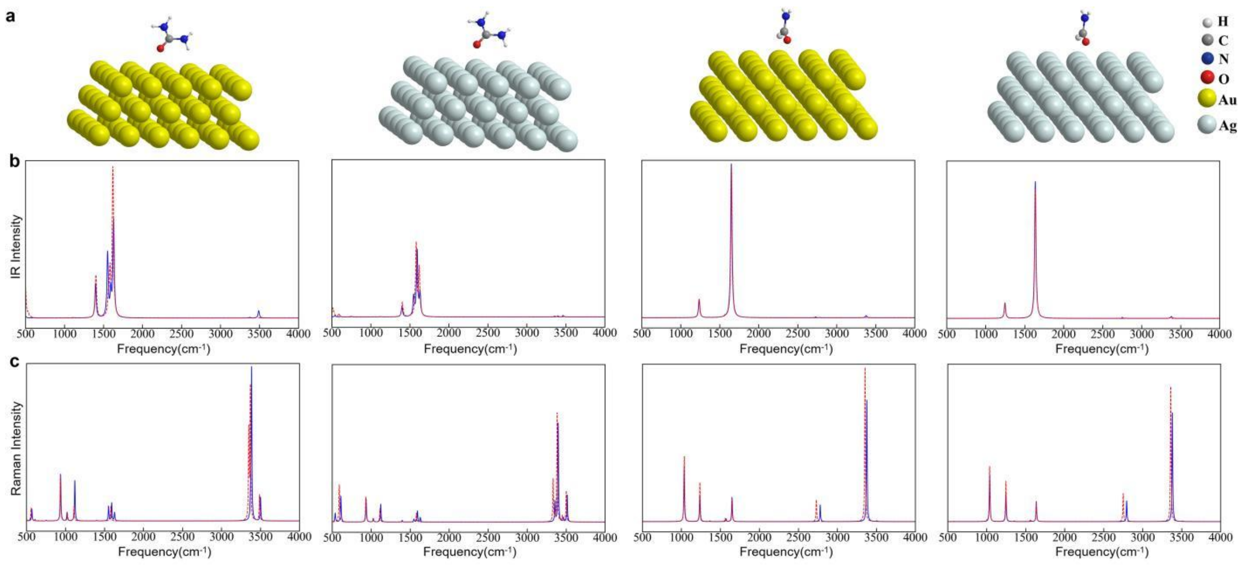
Supplementary Table 5. Comparison of DetaNet with the previously developed methods: Schnet, DimeNet++, PaiNN and Nequip (time is evaluated by training a batch of 64 molecules on an RTX 3080Ti device). Both the prediction error (MAE) for different properties and the computational time were listed. The unit of the NPA charge (q), dipole moment ($\vec{\mu}$), polarizability (α), first hyperpolarizability (β) and computational time (t) is electron (e), Deby (D), a_0^3, a.u. and second (s), respectively. Here N_{RBF} and N_{SHBF} represents the number of radial basis and the number of spherical harmonic basis function, respctively. For the DetaNet and Nequip, e/o is the even/odd parity and 0,1,2,3 is the degree of irreps feature.

Model	MAE (q) Unit: e	MAE ($\vec{\mu}$) Unit: D	MAE (α) Unit: a_0^3	MAE (β) Unit: a.u.	Time (t) Unit: s	Parameter setting
SchNet	0.0034	0.0790	/	/	**0.006**	Number of features = 128; N_{RBF} = 200; Number of interaction layers = 6.
DimeNet++	0.0014	0.0246	/	/	0.268	N_{RBF} = 16; N_{SHBF} = 7; Number of features = 64; Number of interaction layers = 4
Painn	**0.0011**	0.0102	0.1536	/	0.015	N_{RBF} = 16; Number of interaction layers = 4; Number of features = 128; Vector_feature = 3x128.
Nequip	0.0012	0.0124	0.1925	3.3050	0.408	N_{RBF} = 16; Number of interaction layers = 3; Feature = 64x(0e+0o+1e+1o+2e+2o+3e+3o)
DetaNet	**0.0011**	**0.0089**	**0.1367**	**2.0373**	0.027	N_{RBF}=16 Number of interaction layers = 3 Feature=128x(0e+1o+2e+3o)

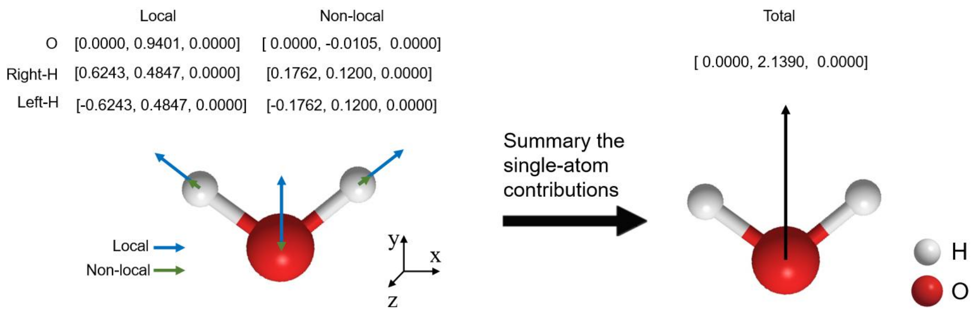
Supplementary Figures



Supplementary Fig. 1|Transferability of DetaNet predictions to molecules of various sizes and novel external environments. a Schematic structures of selected molecules for testing the transferability of DetaNet: lactulose not included in the QM9S, ether molecules in gas phase, in aqueous solution and adsorbed on a gold surface. **b** Spearman correlation coefficients between DetaNet and DFT predictions for IR and Raman spectra of molecules of different sizes. **c** Comparison of DetaNet predicted- and DFT-simulated IR and Raman spectra of lactulose (45 atoms), a molecule not in the QM9S database. **d** Comparison of DetaNet predicted- and DFT-simulated Raman spectra of lactulose. **e** Spearman correlation coefficients between DetaNet and DFT predictions for IR and Raman spectra as a function of degree of molecular deformation induced by an external environment. **f** Comparison of DetaNet predicted- and DFT-simulated IR spectra for ether in the gas phase, in aqueous solution and adsorbed on a gold surface. **g** Comparison of DetaNet predicted- and DFT-simulated Raman spectra for ether in the gas phase, in aqueous solution and adsorbed on a gold surface.

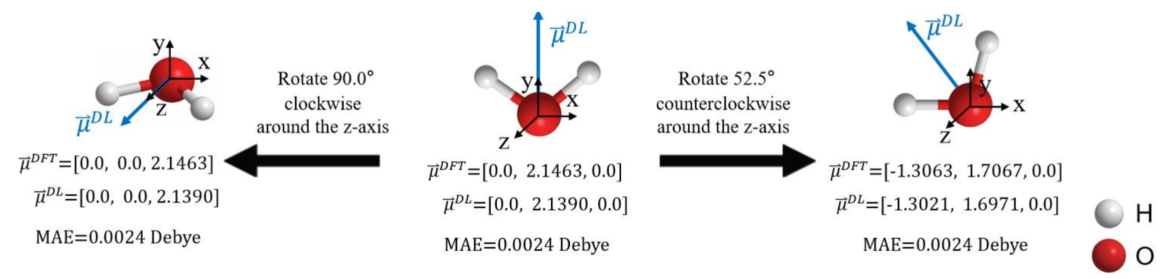


Supplementary Fig. 2| Various examples of different organic molecules adsorbed on metal surfaces. a Structure of urea and formamide molecules adsorbed on Ag(111) and Au(111) surface. **b** Comparison of DetaNet- and DFT-predicted IR spectra for the structures shown above in **a. c** Comparison of DetaNet- and DFT-predicted Raman spectra for the structures shown above in **a.**



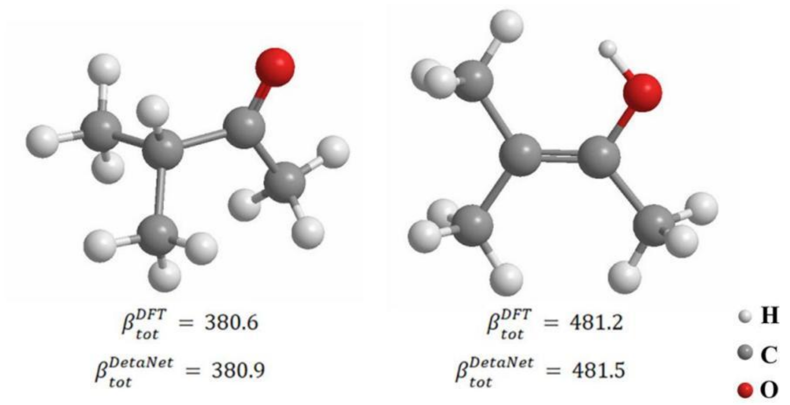
Local
Non-local
Total
O [0.0000, 0.9401, 0.0000] [0.0000, -0.0105, 0.0000]
Right-H [0.6243, 0.4847, 0.0000] [0.1762, 0.1200, 0.0000]
Left-H [-0.6243, 0.4847, 0.0000] [-0.1762, 0.1200, 0.0000]
[0.0000, 2.1390, 0.0000]
Summary the single-atom contributions
Local
Non-local
y
x
z
H
O

Supplementary Fig. 3|Schematic diagram of calculating the total dipole moment for a H_2O molecule based on the local and non-local contributions. The local dipole moment for the oxygen, right and left hydrogen is [0.0, 0.9401, 0.0], [0.6243, 0.4847, 0.0] and [-0.6243, 0.4847, 0.0], while the non-local local parts is [0.0, -0.0105, 0.0], [0.1762, 0.1200, 0.0] and [-0.1762, 0.1200, 0.0]. Summarizing all components up produces the total dipole moment for water is [0.0, 2.1390, 0.0].



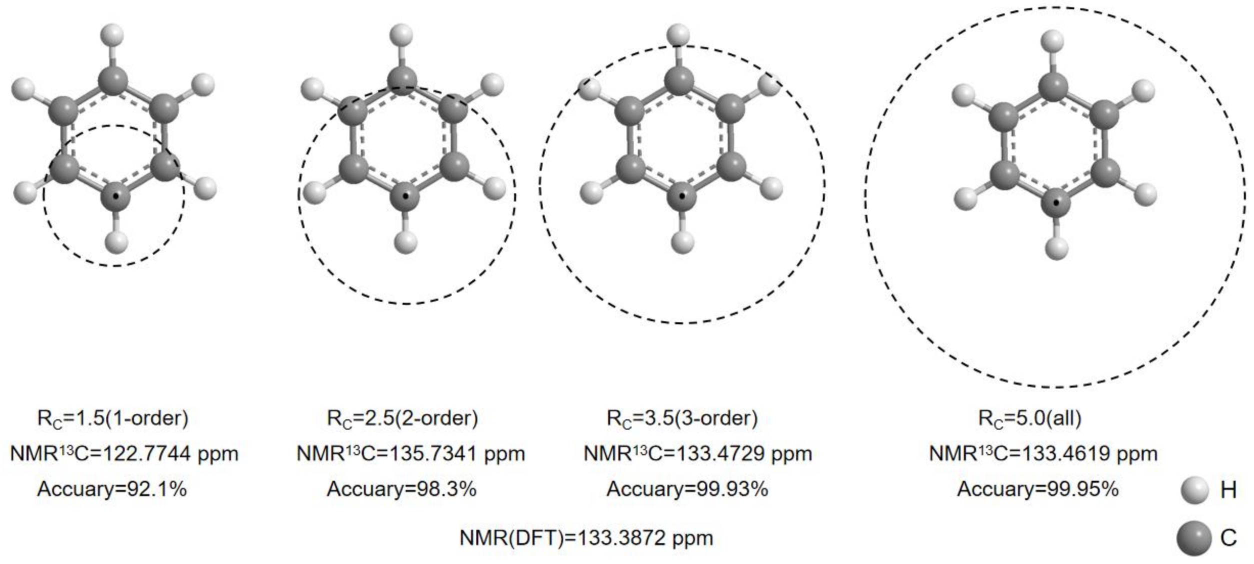
$\vec{\mu}^{DL}$
y
x
z
Rotate 90.0° clockwise around the z-axis
Rotate 52.5° counterclockwise around the z-axis
$\vec{\mu}^{DFT}$=[0.0, 0.0, 2.1463]
$\vec{\mu}^{DL}$=[0.0, 0.0, 2.1390]
MAE=0.0024 Debye
$\vec{\mu}^{DFT}$=[0.0, 2.1463, 0.0]
$\vec{\mu}^{DL}$=[0.0, 2.1390, 0.0]
MAE=0.0024 Debye
$\vec{\mu}^{DFT}$=[-1.3063, 1.7067, 0.0]
$\vec{\mu}^{DL}$=[-1.3021, 1.6971, 0.0]
MAE=0.0024 Debye
H
O

Supplementary Fig. 4|Schematic diagram for the orientation sensitivity of DetaNet. The three-dimensional dipole moment vector of molecules also undergoes corresponding rotation after molecular rotation. MAE represent the mean absolute errors.

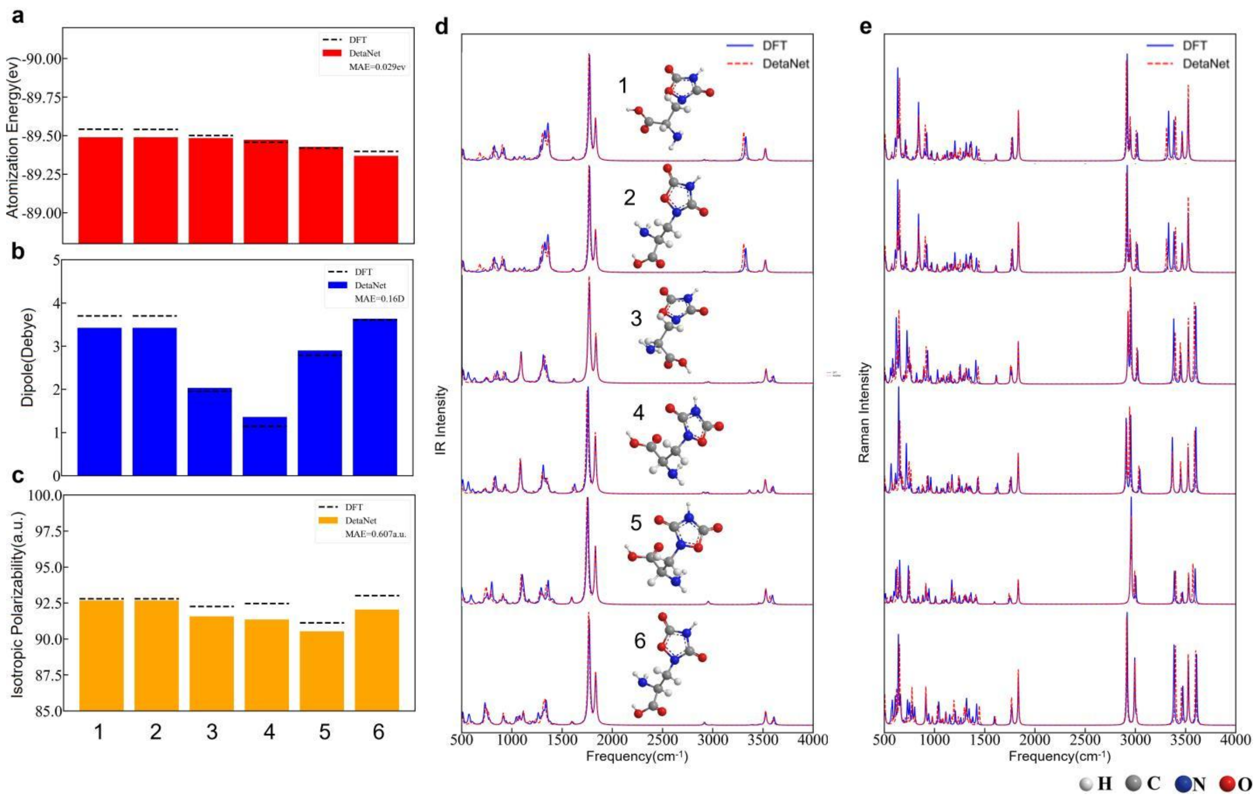


β_{tot}^{DFT} = 380.6
$\beta_{tot}^{DetaNet}$ = 380.9
β_{tot}^{DFT} = 481.2
$\beta_{tot}^{DetaNet}$ = 481.5
H
C
O

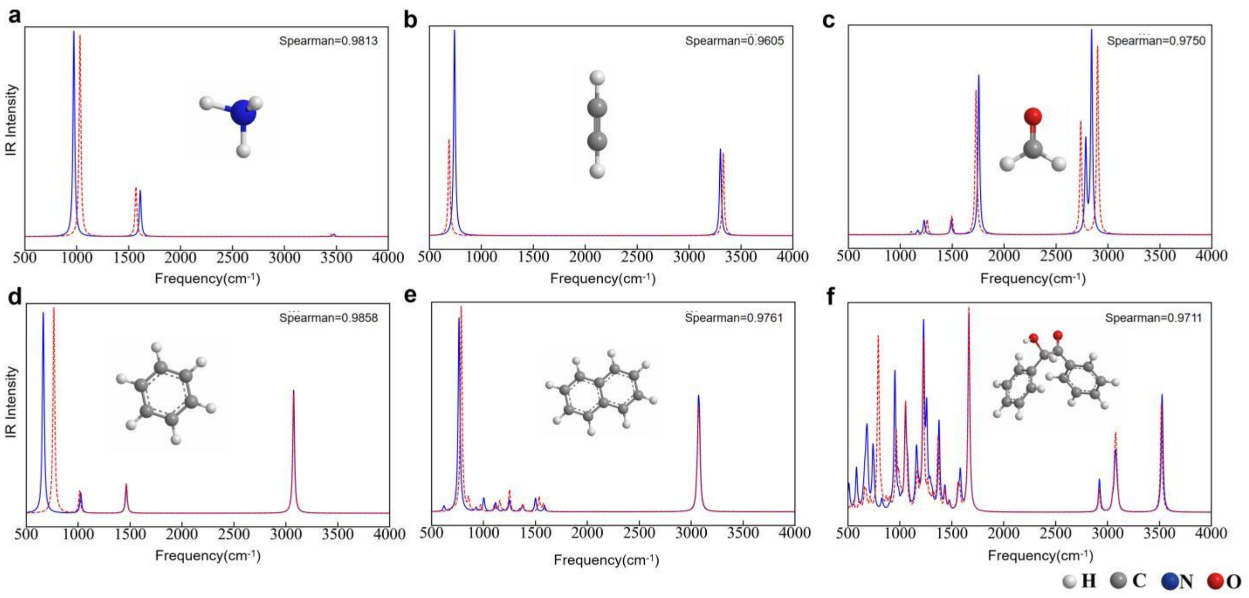
Supplementary Fig. 5| Comparison between the DetaNet and DFT calculated first hyperpolarizability (β) for enol-keto tautomers. Taking the 3-Methyl-2-butanone and 3-Methyl-2-buten-2-ol molecule for example, we found the DetaNet and DFT obtains almost the same first hyperpolarizability with a small deviation of 0.03% and 0.04%.



Supplementary Fig. 6| Comparison between the DFT calculated chemical shift (NMRC) for C atom in benzene molecule and DetaNet calculated result with different cutoff radius (R_c). A contrast experiment on cutoff was performed by setting R_c=1.5, 2.5, 3.5 and 5.0 Å. The set of R_c=5.0 Å indicates the inclusion of the 1st-, 2nd-, 3rd- and 4th-order neighbors and makes the predicted error of DetaNet is only 0.05% compared to DFT.



Supplementary Fig. 7| Performance of DetaNet in simulating flexible molecules. a Comparison of the DetaNet- and DFT- predicted energy**. b** Comparison of the DetaNet- and DFT- predicted dipole moment. **c** Comparison of the DetaNet- and DFT- predicted polarizability. **d** Comparison of the DetaNet- and DFT- predicted infrared spectra. **e** Comparison of the DetaNet- and DFT- predicted Raman spectra**.** The inset molecules named as 1-6 in **d** are the six different conformers of 2-amino-3-(3,5-dioxo-1,2,4-oxadiazolidin-2-yl) propanoic acid. MAE represent the mean absolute errors.



Supplementary Fig. 8| Illustrated possible pitfalls of DetaNet. **a** Comparison of DetaNet- and DFT-predicted IR spectra for ammonia. **b** Comparison of DetaNet- and DFT-predicted IR spectra for acetylene. **c** Comparison of DetaNet- and DFT-predicted IR spectra for formaldehyde. **d** Comparison of DetaNet- and DFT-predicted IR spectra for Benzene. **e** Comparison of DetaNet- and DFT-predicted IR spectra for naphthalene. **f** Comparison of DetaNet- and DFT-predicted IR spectra for benzoin. Spearman coefficients is used to assess the similarity between the DetaNet- and DFT-predicted spectra.

References

1 Nagelkerke, N. J. A note on a general definition of the coefficient of determination. *Biometrika* **78**, 691-692 (1991).

2 Zar, J. H. Significance testing of the Spearman rank correlation coefficient. *Journal of the American Statistical Association* **67**, 578-580 (1972).

3 Westermayr, J., Gastegger, M. & Marquetand, P. Combining SchNet and SHARC: The SchNarc machine learning approach for excited-state dynamics. *The journal of physical chemistry letters* **11**, 3828-3834 (2020).

4 Westermayr, J. & Marquetand, P. Deep learning for UV absorption spectra with SchNarc: First steps toward transferability in chemical compound space. *The Journal of Chemical Physics* **153** (2020).

5 Ye, S. *et al.* A neural network protocol for electronic excitations of N-methylacetamide. *Proceedings of the National Academy of Sciences* **116**, 11612-11617 (2019).

6 Wang, Y. *et al.* PubChem: a public information system for analyzing bioactivities of small molecules. *Nucleic acids research* **37**, W623-W633 (2009).

7 Makov, G. & Payne, M. C. Periodic boundary conditions in ab initio calculations. *Physical Review B* **51**, 4014 (1995).

8 Hafner, J. Ab-initio simulations of materials using VASP: Density-functional theory and beyond. *Journal of computational chemistry* **29**, 2044-2078 (2008).

9 Schütt, K. T., Sauceda, H. E., Kindermans, P.-J., Tkatchenko, A. & Müller, K.-R. Schnet–a deep learning architecture for molecules and materials. *The Journal of Chemical Physics* **148** (2018).

10 Gasteiger, J., Giri, S., Margraf, J. T. & Günnemann, S. Fast and uncertainty-aware directional message passing for non-equilibrium molecules. *arXiv preprint arXiv:2011.14115* (2020).

11 Schütt, K., Unke, O. & Gastegger, M. in *International Conference on Machine Learning.* 9377-9388 (PMLR).

12 Batzner, S. *et al.* E (3)-equivariant graph neural networks for data-efficient and accurate interatomic potentials. *Nature communications* **13**, 2453 (2022).