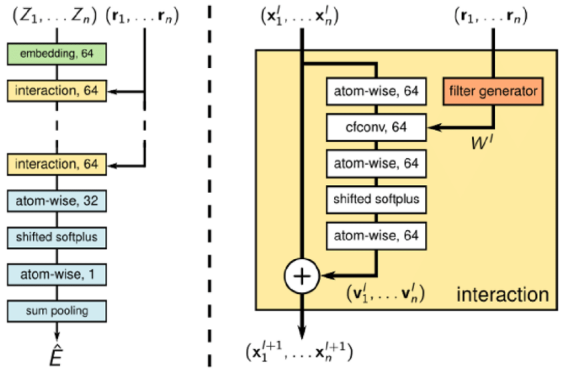
Molecular Property Prediction Challenge

Model Architecture

SchNet is a variant of deep tensor neural networks (DTNNs) that inherently respects the fundamental symmetries of atomistic systems, such as rotational and translational invariance, as well as invariance to the ordering of atom indices. Its architecture utilizes continuous-filter convolutional layers to effectively capture both spatial and chemical interactions. The model represents each atom as a feature vector $x^{(l)}_i \in R^D$, where D denotes the feature space dimension and l represents the layer in the network. Atomic interactions are iteratively updated T times through pairwise interactions between feature vectors of atoms within a defined cutoff distance, incorporating information about the chemical environment and complex many-body interactions. The continuous-filter convolution layers, facilitated by filter-generating networks, refine these feature representations. Finally, the model pools atomwise updates to predict global molecular properties, ensuring an accurate and efficient mapping of structure-property relationships.



Why choose PaiNN over SchNet?

We employed a polarizable atom interaction neural network (PaiNN) to predict accurate dipole moments. PaiNN is particularly effective for predicting dipole moments because it learns from both geometric and directional information in molecular structures. It processes not only the positions of atoms but also how their spatial interactions influence directional properties like dipole vectors. By preserving rotation equivariance, PaiNN ensures that if a molecule is rotated, the predicted dipole moment rotates accordingly therefore aligning with physical reality. Its message-passing layers incorporate 3D structural and angular information, enabling accurate modeling of charge separation and molecular polarization, which are critical for calculating dipole moments.

While SchNet is better than grid-based architectures which can cause changes in atomic properties based on position of the atom in the grid, SchNet uses continuous convolutional layer to be translation and rotation invariant. This architecture works well for scalar properties. But for vector properties such as dipole moments, the model needs to be

equivariant which means it should depend upon rotation etc. PaiNN incorporates not only interatomic distances but also atomic charges and directional (vector) embeddings, which is advantageous for the prediction of dipole moments.

Hyperparameter Optimization

Optimization of hyperparameters in SchNet is a tedious process as numerous layers and corresponding variables are involved. To facilitate this process, we used Optuna to optimize relevant hyperparameters to minimize the loss function.

```
Best value: 0.08051550388336182
Best params: {'bs': 400, 'lr': 0.0005, 'patience': 13, 'n_rbf': 17, 'n_interactions': 7, 'n_atom_basis': 128, 'cutoff': 6, 'pat2': 11}
   number     value            datetime_start         datetime_complete
0       0  0.300311 2025-04-05 04:06:35.153861 2025-04-05 05:05:03.701233  \
1       1  0.497945 2025-04-05 05:05:03.719607 2025-04-05 06:03:59.844491
2       2  0.309109 2025-04-05 06:03:59.858613 2025-04-05 07:02:33.315416
3       3  2.011197 2025-04-05 07:02:33.329626 2025-04-05 08:00:59.659471
4       4  1.159986 2025-04-05 08:00:59.672963 2025-04-05 09:00:14.297059
5       5  0.359730 2025-04-05 09:00:14.311220 2025-04-05 09:59:09.748421
6       6  1.165218 2025-04-05 09:59:09.762201 2025-04-05 10:59:01.582087
7       7  0.080516 2025-04-05 10:59:01.596271 2025-04-05 11:58:26.268182
8       8  0.294254 2025-04-05 11:58:26.282486 2025-04-05 12:57:50.152654
9       9  0.341211 2025-04-05 12:57:50.166296 2025-04-05 13:56:22.243974
```

Model Training and Prediction

Using the hyperparameters obtained from Optuna, we trained the model with a walltime of approximately 2 hours on a single Nvidia RTX A6000 GPU. To enhance generalizability and prevent overfitting, we employed an EarlyStopping callback based on the validation loss. The model was evaluated using Mean Absolute Error (MAE) and Root Mean Squared Error (RMSE) as performance metrics. Training progress was monitored and logged using TensorBoard.

```python
cutoff = 6.
n_atom_basis = 128

pairwise_distance = spk.atomistic.PairwiseDistances()
radial_basis = spk.nn.GaussianRBF(n_rbf=17, cutoff=cutoff)
painn = spk.representation.PaiNN(
    n_atom_basis=n_atom_basis, n_interactions=7,
    radial_basis=radial_basis,
    cutoff_fn=spk.nn.CosineCutoff(cutoff),
    activation=F.silu
)


pred_correction = spk.atomistic.Atomwise(n_in=n_atom_basis, output_key='dpm')

nnpot = spk.model.NeuralNetworkPotential(
    representation=painn,
    input_modules=[pairwise_distance],
    output_modules=[pred_correction],
    postprocessors=[
        spk.transform.CastTo64(),

    ]
)

output_corr = spk.task.ModelOutput(
    name='dpm',
    loss_fn=torch.nn.MSELoss(),
    loss_weight=1.,
    metrics={
        'MAE': torchmetrics.MeanAbsoluteError(),
        'RMSE': torchmetrics.MeanSquaredError()
    }
)
```

```python
task = spk.task.AtomisticTask(
    model=nnpot,
    outputs=[output_corr],
    optimizer_cls=torch.optim.AdamW,
    optimizer_args={'lr': 5e-4},
    scheduler_cls=spk.train.ReduceLROnPlateau,
    scheduler_monitor='val_loss',
    scheduler_args={'mode': 'min', 'factor': 0.5, 'patience': 11, 'threshold_mode': 'rel', 'cooldown': 5},
)

logger = pl.loggers.TensorBoardLogger(save_dir=new_datatut)
callbacks = [
    spk.train.ModelCheckpoint(
        model_path=os.path.join(new_datatut, f'best_inference_model_fold'),
        save_top_k=1,
        monitor='val_loss'
    ),
    pl.callbacks.EarlyStopping(monitor='val_loss', mode='min', patience=13, min_delta=0),
    pl.callbacks.LearningRateMonitor(logging_interval='step'),
    spk.train.ExponentialMovingAverage(decay=0.995)
]
```

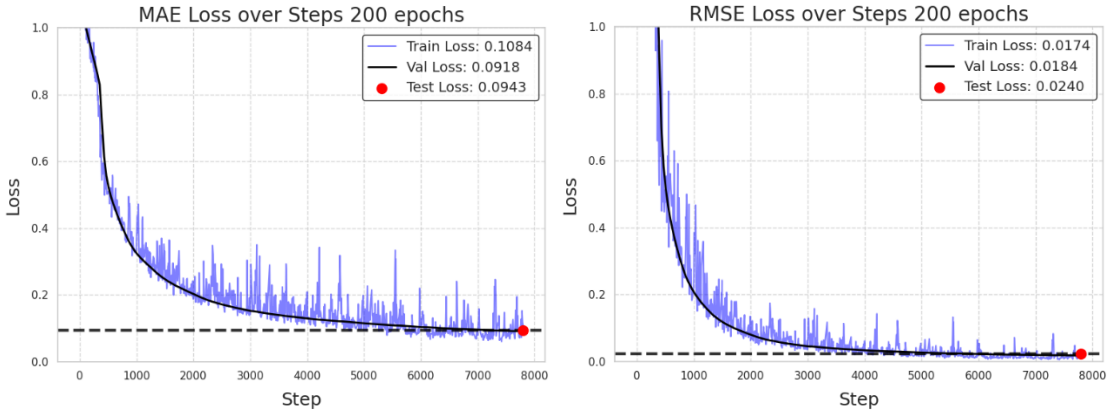
```python
trainer = pl.Trainer(
    max_epochs=200,
    callbacks=callbacks,
    default_root_dir=new_datatut,
    accelerator='auto',
    devices='auto',
    accumulate_grad_batches= 1,
    val_check_interval= 1.0 ,
    check_val_every_n_epoch= 1 ,
    num_sanity_val_steps= 0,
    fast_dev_run= False,
    enable_checkpointing=True,
    overfit_batches= 0 ,
    limit_train_batches= 1.0  ,
    limit_val_batches= 1.0  ,
    limit_test_batches= 1.0,
    log_every_n_steps=5

)

trainer.fit(task, datamodule=newdata)
dataloaders=newdata.test_dataloader()
trainer.test(task, dataloaders=dataloaders)
print('training complete!')
```

Test metric	DataLoader 0
test_dpm_MAE	0.09429313242435455
test_dpm_RMSE	0.02400350384414196
test_loss	0.02400350384414196

training complete!



The model showed consistent performance on the training, validation, and test sets, indicating strong generalization without overfitting. By combining SchNet's ability to model molecular environments using continuous filters instead of fixed grids, with PaiNN's sensitivity to 3D geometry and directional interactions, our approach is well-suited for predicting complex properties like dipole moments.