School of Chemistry

Final Year Project 2024-25

(Module CH3325)

Investigating Generalisation Limits of Deepset Architectures for Molecular Energy and Force Predictions

Harry George Badland

Supervisor: Dr Stefano Leoni

Contents

List of Figures

List of Tables

1 Abstract

1.1 Abstract

This project investigates architectural modifications to the deepset model within TorchMD with the intent of improving molecular energy and force predictions. Scalar feature augmentations, specifically through the concatenation of non-linear representations, were explored to improve atomic representations. The integration of graph neural network (GNN) layers such as SAGEConv, GATConv, and EDGEConv was also assessed, followed by the expansion of the radial basis function (RBF) using similar non-linear attributes. Evaluation on but-2-ene showed that cubic and square root augmentations improved the accuracy of energy prediction by 39.1%, while GNN additions and RBF transformations consistently reduced accuracy, with the former exhibiting a minimum accuracy reduction of 17% due to oversmoothing node representations. However, evaluation on unseen pentene conformers revealed notable performance degradation, consistently underestimating energies by over 3 hartree, which highlighted the challenges with generalisation in machine learning interatomic potentials. The analysis of per-atom energy predictions detailed that the deepset model learned interpolative patterns that were limited to the training distribution, with poor adaptability to novel structures. These findings suggest that enhancing extrapolative capabilities in molecular machine learning will require both broader chemical training datasets and more physically informed model designs.

1.2 Acknowledgements

I would like to extend my gratitude to Masoud Nourallah for his constant support and guidance throughout this project. Masoud taught me a lot about coding over so many hours and made the project thoroughly enjoyable. I have enjoyed each day spent working on it under his guidance.

I would also like to thank Dr Stefano Leoni for his exceptional support and supervision, providing me with a project which I found extremely engaging and always making time to offer support.

2 Introduction

2.1 Neural Networks

A neural network is a framework for function approximation that mimics the human brain's ability to learn complex functions. Neural networks consist of interconnected nodes (neurons) that process data and learn patterns, enabling them to complete tasks such as classification or regression. Each neuron applies an activation function, which allows the model to handle more complex relationships by introducing nonlinearity.[1;2] Neural networks have been a topic of research since 1943, when McCullough and Pitts published the first mathematical model, building upon ideas already introduced by Alan Turing.[3] They were first employed in chemistry in late 1980s, where they were used for establishing relationships between molecular structure and spectra and for the prediction of chemical and physical properties.[4] However, recently, the development of AlphaFold(2), a model developed by Isomorphic labs and backed by Google has demonstrated the increasing influence of computational chemistry. AlphaFold(2) is a neural network that uses transformers to predict protein structures with unprecedented accuracy, and was recognised with the 2024 Nobel Prize, showing just how relevant and progressive the applications of neural networks in chemistry are, especially in the direction modern chemistry is heading.[5] Mathematically, a neural network can be expressed as:

$$y = f(Wx + b) \tag{1}$$

Where x is the input vector, W and b are learnable parameters representing weights and biases respectively and f is a non-linear activation function, (commonly ReLU or sigmoid).[2] In an adaptive learning environment, such as the deepset model in this paper, the weights and biases are updated based on the error gradients computed by back-propagation, mathematically given by:

$$W^{(l)} = W^{(l)} - \eta \nabla_{W^{(l)}} L \tag{2}$$

Where $\nabla_{W^{(l)}} L$ is the gradient of the loss function L with respect to the weights in layer l. The learning rate (η), determines the magnitude of these updates, is automatically adjusted accordingly to balance convergence speed with stability, ensuring that the model does not overshoot optimal solutions.[6]

Multiple activation functions are used in computational chemistry, in this study SiLU, given below, was used because it is advantageous in molecular modelling due to being more stable.[7]

$$\text{SiLU}(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}} \tag{3}$$

SiLU prevents the output from growing too large, which limits the common issue of exploding gradients that destabilise the training process, especially in such large datasets, such as but-2-ene and pent-2-ene which are described later on in Section 3.1.[8] Training a neural network involves feeding data forward into the network and calculating the error via a loss function, which is then used to compute the weights to minimise this error. There are many optimisation algorithms, notably SGD (stochastic gradient descent), with Adam which are employed in deepset.[9;10]

2.2 Transformers

In neural networks, a transformer, initially proposed in 2017 by Vaswani *et al*, is a novel architecture, designed to learn context while handling long-range dependencies with ease. Most notably, transformer models are the foundational architecture behind large language models (LLMs) such as the popular generative pre-trained transformer (GPT), highlighting their prevalence and ever-increasing impact in all fields, not just computational chemistry.[11;12]

Although originally developed for natural language processing (NLP), transformers are a key tool used in molecular modelling, particularly drug discovery.[13] Their success in computational chemistry can be attributed to the mathematical attention mechanism, with the model consisting of stacks of encoders and decoders working on the mechanism in order to accurately learn the context behind long-range dependencies by weighing the influence of different input values when producing an output. Alongside the attention mechanism, the adaptability of transformers, particularly in being able to handle diverse data applications such as molecular graphs, is vital in molecular chemistry because the natural data structure for the representation of molecules are graphs.

Since the first application in 2017, many models have been created that utilise transformers in computational chemistry, notably the intermolecular graph transformer (IGT), which employs a dedicated attention mechanism to model intermolecular information, consisting of a feature extraction module, a message passing module, and a readout module. This model was created to provide a solution to the costly and time ineffective *de-novo* drug design by screening from known molecules. This method is applied in both binding activity and binding pose predictions and is proving to be successful when tested on real-world datasets in DTI scenarios.[14]

In the TorchMD architecture and deepset, an equivariant transformer, which ensures the model predictions change consistently when the input molecule is rotated is used. Notably, an embedding layer encodes the atom type (Z), and the neighbourhood of each atom into a feature vector (x_i), followed by multiple update layers which compute the interactions between atom wise pairs, eventually resulting in a single molecular prediction (final output) as illustrated in Figure 1.[15]

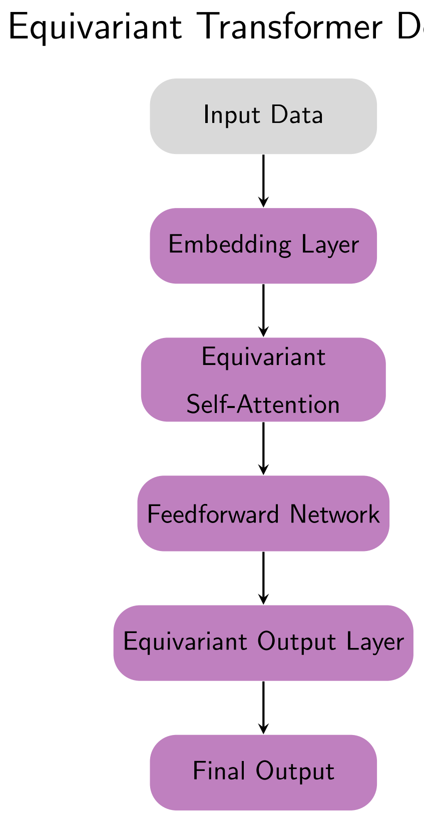


Figure 1: Flow chart of the transformer model in deepset.

2.3 Convolutional Layers

Convolutional layers are the main building block of a convolutional neural network (CNN), they contain a set of kernels that slide over the data, perform element-wise multiplications and subsequently produce an activation map.[16] In computational chemistry, convolutional layers are used since they specialise at extracting spatial features from data, such as the grid-based molecular representations seen in molecular modelling. Recently, CNNs have been applied to predict gaseous adsorption properties in metal–organic frameworks (MOFs) by encoding atomic positions and chemical information.[17] This is similar to the proposed application of CNNs in deepset, which encodes atomic positions amongst a range of spatial features to improve energy and force predictions, meaning that there is a proven theoretical basis for their implementation.

Despite there being more than 30 different types of convolutional layer in the PyTorch Geometric (PyG) package, four different convolutional layers were used in this project to gain an understanding of whether they improve the model's predictions. SAGE is a variant of GraphSAGE, which differs slightly in the aggregation function used. The convolutional layer, SAGE accounts for the degrees of nodes within the neighbourhood, which is effective in molecular dynamics, since it can capture more detailed structural information. The algorithm is inspired by the Weisfeiler-Lehman Isomorphism Test, a classic algorithm for graph isomorphism. SAGE approximates this test using trainable aggregators to generate node embeddings. In comparison to GCNs (graph convolutional networks), which are limited to only taking the mean of neighbours, SAGEConv outperforms since it can use varying aggregation strategies such as pooling, which can be interchanged to improve accuracy by allowing more control, resulting in a more suitable layer for handling large-scale molecular graphs.[18] The mathematical operator is shown below:

$$x'_i = W_1 x_i + W_2 \cdot \text{mean}_{j \in N(i)} x_j \tag{4}$$

In contrast, graph attention convolution (GAT) layers are another convolutional layer which is prevalent in deep learning for molecular dynamics. GAT layers focus on relevant parts of the graph structure by utilising a self-attention mechanism where nodes assign their neighbours different weightings. This is an important feature for molecular dynamics because of the number of forces involved and the importance of spatial features when computing molecules. In a study by Stanford in 2020, GAT was shown to increase toxicity prediction accuracy by over 5% compared to other standard graph convolutions.[19] This suggests that the deepset model could benefit from the attention mechanism for predicting energies and forces. GAT is mathematically defined as follows:

$$x'_i = \sum_{j \in N(i) \cup \{i\}} \frac{1}{\alpha_{i,j}} \Theta_t x_j \tag{5}$$

EDGE, on the other hand, is a deep learning algorithm for processing point cloud data which is useful in molecular modelling since it focuses on the relationships between neighbouring nodes (atoms) by explicitly using edge features. It is commonly applied in GNNs to extract local structural features whilst preserving spatial relationships between atoms, instead of training atomic features independently, EDGE computes edge-dependent features using a function of both an atom and its neighbours, as outlined below:

$$x'_i = W_1 x_i + W_2 \cdot \text{mean}_{j \in N(i)}(x_j) \tag{6}$$

When implementing a GNN within Deepset, it is important to consider different types of convolutional layer, as SAGE, GAT, and EDGE each handle atomic representations differently. Accounting for these variations enables a more comprehensive representation of atomic interactions, which can improve the accuracy of the model.

2.4 TorchMD

TorchMD is an open source framework built to take advantage of the primitives of PyTorch. First published by Doerr *et al* in 2021, it was designed specifically for the simulation of molecular dynamics and is popular for integrating deep learning techniques to accelerate molecular simulations, providing a reliable cost-effective alternative to techniques such as DFT.[20] TorchMD enables the integration of machine learned potentials by extending bonded and non bonded force terms commonly used in molecular dynamics. In this project, I implement all changes and additions into the deepset. Deepset processes individual atomic environments and aggregates them using permutation invariant operations to produce energy and force predictions. It embeds atom types, computes distance-based features using radial basis functions, and models interatomic interactions via a gating mechanism over a set of expert networks.

The framework allows machine learning modules such as graph neural networks (GNN) to be implemented on top, enhancing representation of classical force fields with learnt potentials that capture more complex interactions and implements the functional form of the AMBER potential.[21] This approach accounts for all terms including angles, torsions, electrostatic energies, and non bonded van der Waals forces, as well as allowing the use of other external potential energies, V_{ext}. The total potential energy is calculated from the equation described in the following.[22;23]

$$V_{total} = \sum_{n_{bonds}} V_{bonded} + \sum_{n_{angles}} V_{angle} + \sum_{n_{torsions}} V_{torsion} + \sum_{i}^{n_{atoms}} \sum_{k<i}^{n_{atoms}} (V_{vdw} + V_{electrostatic}) + \sum_{n_{ext}} V_{ext} \quad (7)$$

Recently, TorchMD has emerged as a more popular framework for molecular dynamic (MD) simulations, particularly for simulating protein folding processes. One notable application involves the CLN025 variant of Chignolin, a 10-residue β-hairpin miniprotein, Figure 2, whose folding occurs on nanosecond timescales. TorchMD's coarse grained (CG) neural network potential (NNP) reduces computational demand by representing groups of atoms as single interaction sites, enabling efficient simulation of large biomolecules. TorchMD-Net trained a SchNet based model using force matching on approximately 9 milliseconds of MD data to replicate the folding thermodynamics of CLN025.[24] This approach achieved a 50-fold increase when compared to conventional all atom simulations, while maintaining folding pathway accuracy. The two main simulation types in TorchMD are the CG model and the all-atoms model (AA). The CG model was chosen to simulate protein folding since it allows access to longer timescales and is computationally more efficient, since it groups multiple atoms in a single bead, whereas the all atoms model requires the forces for every single atom to be calculated at each timestep, which is highly demanding.[23]

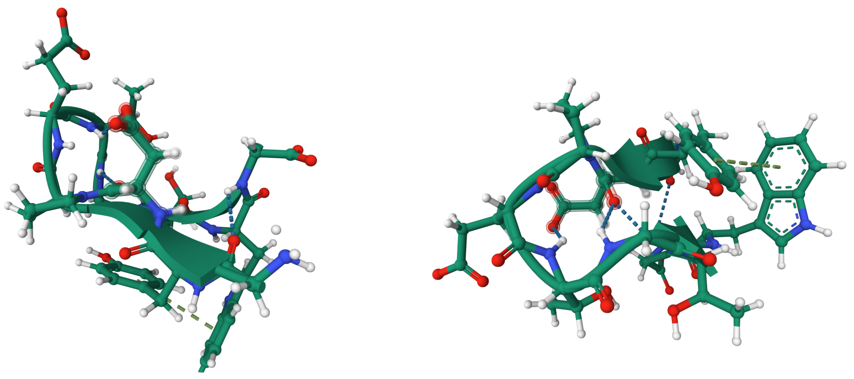


Figure 2: 3D structure of chignolin protein.

The integration of PyTorch's automatic differentiation (autograd) in TorchMD allowed for gradient based optimisation of NNPs, where the loss gradient is given by:

$$\frac{\partial \text{Loss}}{\partial W} = \frac{\partial \text{Loss}}{\partial a} \cdot \frac{\partial a}{\partial z} \cdot \frac{\partial z}{\partial W} \tag{8}$$

Although PyTorch handles this automatically, this equation can be written more explicitly as:

$$\frac{\partial \text{Loss}}{\partial W} = \delta \cdot x^T \tag{9}$$

Where:

$$\delta = \frac{\partial \text{Loss}}{\partial a} \cdot \frac{\partial a}{\partial z} \tag{10}$$

and x is the input to the layer, while δ is the error term from the backpropogation of the loss.

By automatically calculating the gradient of the loss function, autograd updates the weights using optimisers such as SGD or Adam (implemented in deepset). This technique demonstrated energy conservation errors as low as 1.1×10^{-5} K per degree of freedom in validation runs. These computational advances position TorchMD as a critical tool for bridging machine learning and physics-based modelling in protein folding studies.[25] The two main types of simulation in TorchMD are the coarse-graining model (CG) and the all-atoms model (AA). The CG model was chosen to simulate protein folding, because it allows access to longer timescales and is computationally more efficient, since it groups multiple atoms in a single bead.

More recently, an updated TorchMD-Net 2.0 has been proposed, which exhibits significant advances in computational efficiency and modular design. One major innovation is the addition of TensorNet, an O(3)-equivariant message-passing neural network, that achieves both high accuracy and an increase in computational efficiency. By adding CUDA graphs and optimising neighbour search algorithms (which was previously a restriction), Torch-MD-Net 2.0 provides up to 10 fold performance improvements when compared to previous versions, outlining just how fresh this architecture is and how constant developments are being published in this field.[26]

3 Model Environment

3.1 Computational Environment and Reproducibility

All experiments were carried out using the TorchMD-Net framework implemented in Python 3.12.9. The model was trained using PyTorch 2.6.0, since all experiments were conducted on an M4 Pro Apple Macbook, CUDA was not used and instead the model was trained in CPU mode. Logging was performed using Weights & Biases and TensorBoard, allowing for experiment tracking and comparison across model variants. Random seeds were fixed across NumPy and PyTorch, to maintain deterministic behaviour. The model demonstrated consistent convergence within 150 epochs for each variation, so remained constant during training.

The datasets used in this project contain over 10,000 conformers for but-2-ene and pent-2-ene, they each contain all possible conformer types, ranging from E to Z isomers, with their energies recorded in Hartree. But-2-ene was selected as the training molecule for its structural complexity and range of interatomic forces. The conjugated alkene backbone influences energies with internal degrees of freedom, angles, torsions and exhibits sensitivity to both short and long range interactions. These factors make but-2-ene a complex molecule for training deepset, but also one which is chemically rich and can be used to effectively test generalisation. The choice of pent-2-ene is similar to but-2-ene, however, the additional carbon tests how the model can adapt its predictions while keeping the types of interactions constant, providing greater as opposed to testing on a structurally different molecule. Each of these datasets contain over 10,000 conformers and are available in the github (below) as both raw (.xyz) and processed (.pt) files.

The model history and all changes are available at https://github.com/hbadland/Torch-MD-net/tree/harry-project.

3.2 Model Architecture and Configuration

This implementation extends the TorchMD framework with a novel deepset model, which passes atomic features through a series of transformations (d_ij_transform, a_i_transform and a_j_transform) to prepare node and edge features for message passing between atoms. The deepset class employs a mixture of experts (MoE) mechanism, which enables the model to specialise different experts to capture diverse types of atomic interactions, enhancing the flexibility of the model. The gating mechanism, which uses a distance aware softmax function to select and weigh the top 5 contributors for each interaction (k = 5) activates the most relevant experts based upon the learned atomic environment, improving the representation without incurring extra computational demand.[27]

The atom and pairwise features are represented by a multi layer embedding system which enables context aware feature interactions throughout the network, providing a comprehensive understanding of the atomic (node) representations within the conformers. Atom types are embedded via a fixed size linear layer, while interatomic distances are encoded using a radial basis function (RBF) with a Gaussian kernel.[28] The outline of how the deepset model processes data is given below, in Figure 3.

The deepset model is configured through a yaml file, given below as Figure 4 , which enables for quick adjustment over model parameters and training conditions such as epochs, batch size and cutoffs. The base and outer cutoff are defined at 0 Å and 5 Å respectively, meaning only atoms within a radius of 10 Å or below are considered to have an electronic contribution, this was chosen since in molecular systems, the influence of interatomic forces decrease with distance, although the message passing mechanism will push the boundaires of this. By setting vector_cutoff to "True", we ensured all directional information is included,

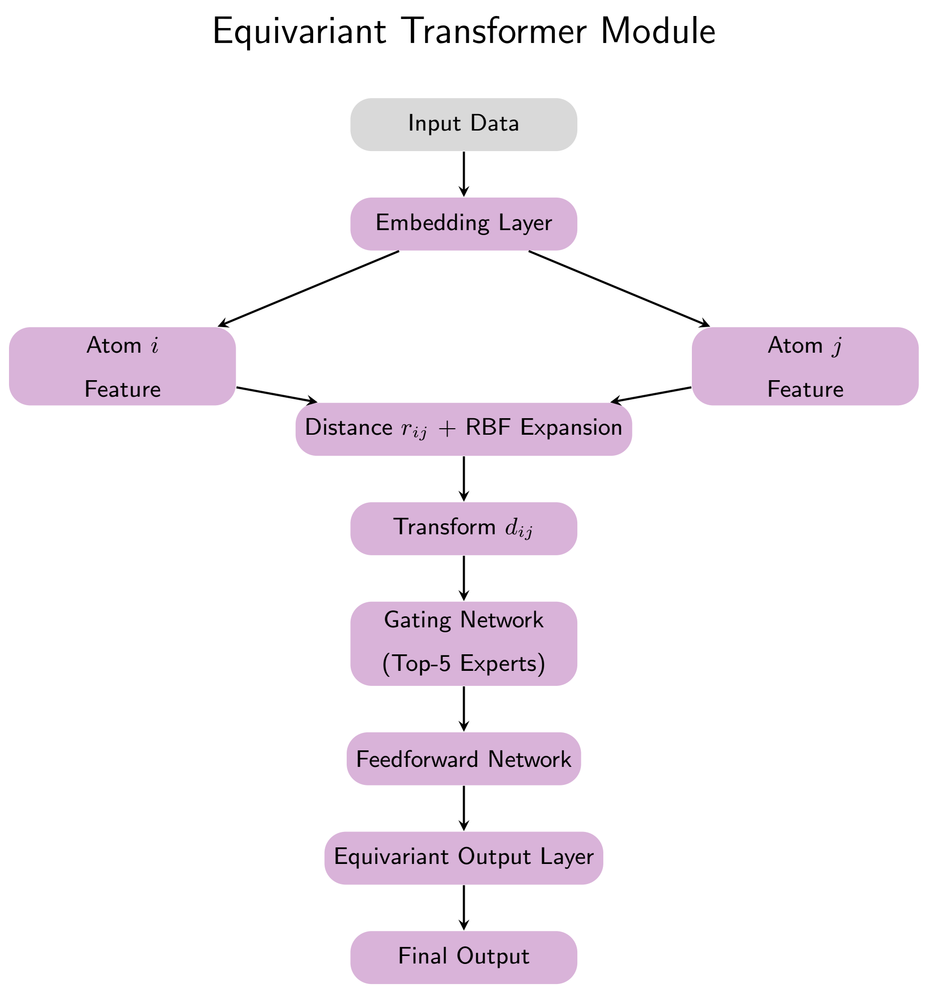


Figure 3: Flow chart of the TorchMD deepset message passing.

which is vital when computing conformers for message passing. The embedding dimension, which refers to the vector size that represents each atom, was set to 128 to maximise the trade-off between computational cost and model accuracy, allowing for a greater capacity of the learned atomic representations at moderate computational cost. Similarly, the model uses 32 RBFs with expnorm distribution to encode interatomic distances. These embeddings are set to not be trainable to help limit overfitting by reducing the number of trainable parameters.

```

    model: deepset
    embedding_dimension: 128 # Size of vector that represents each atom
    expert_out_features: 128 # Number of output features from MoE
    num_layers: 6 # Number of message passing layers
    num_gates: 10 # Number of gates in MoE gating mechanism
    top_k: 5 # Number of relevant experts selected per input
    activation: silu # Activation function type
    aggr: add # Method to combine messages
    rbf_type: expnorm # Normalised exponential RBF
    num_rbf: 32 Number of RBF to encode distances
    trainable_rbf: false # RBF not learnable (fixed)
    base_cutoff: 0.0 # Base cutoff not used in deeepset, no minimum
    outer_cutoff: 10.0 # Cutoff for atomic interactions
    vector_cutoff: true # Applies cutoff to directional features

    dataset: MDCustomInMemory
    dataset_root: ~/data
    train_size: 1000 # Number of training samples
    val_size: 100 # Number of validation samples
    test_size: 100 # Number of validation samples
    derivative: true # Train on force labels as well as energy

    batch_size: 20 # Number of samples per batch in training
    num_epochs: 150 # Total training epochs
    lr: 0.001 # Initial learning rate
    lr_warmup_steps: 1000 # Number of steps to stabilise lr initially
    lr_patience: 30 # If loss doesnt improve for 30 epochs then adjust
    lr_factor: 0.2 # Factor by which lr adjusts
    neg_dy_weight: 0.8 # Weighting factor for force prediction loss
    y_weight: 0.2 # Weighting factor for energy prediction loss

```

Figure 4: Sample of code outlining the key parameters during training and their meaning.

Optimisation was performed using the Adam optimiser, a variant of the PyTorch module, stochastic gradient descent (SGD) designed to provide stable convergence.[10] Adam was employed for its adaptive learning rate mechanism that improves convergence speed and robustness to local minima, which is especially relevant for conformational datasets such as but-2-ene. The process of how Adam works is shown in Figure 5.

4 Training and Results

The metrics used for evaluating the performance of the model are the force loss and energy loss. Specifically we are considering the averages of the test and validation values for each metric, because the validation set was not used for hyper parameter optimisation it cannot be regarded as a legitimate validation set in this context, thus it is more accurate to consider the average. The force loss is considered a more relevant metric for the analysis as it is calculated from a larger number of predicted values, providing a more thorough

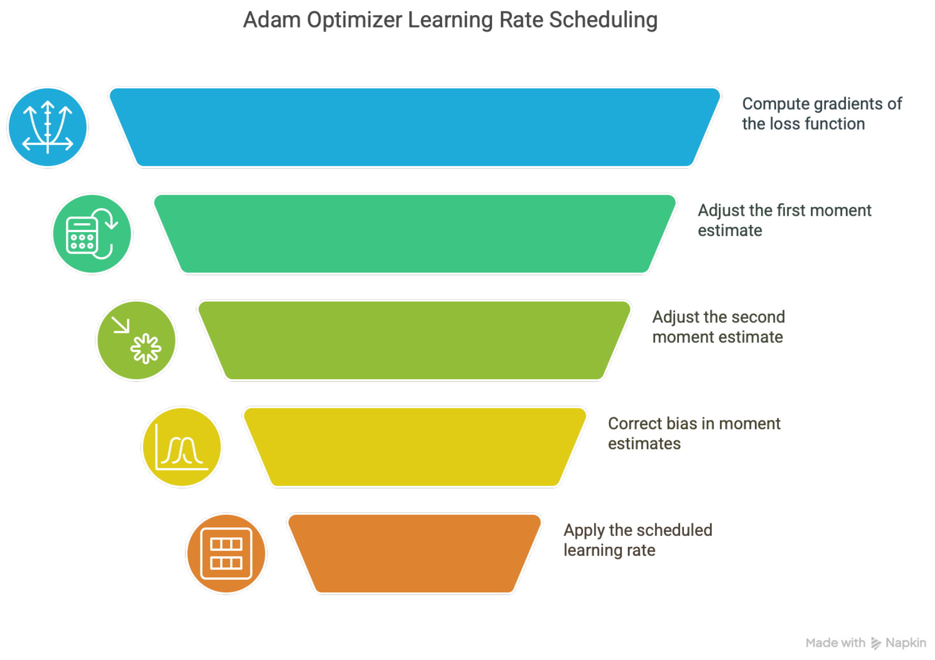

Adam Optimizer Learning Rate Scheduling
Compute gradients of the loss function
Adjust the first moment estimate
Adjust the second moment estimate
Correct bias in moment estimates
Apply the scheduled learning rate
Made with Napkin

Figure 5: Visual representation of the Adam optimisation algorithm used during model training.

assessment of model performance.

The energy loss for the model is calculated using a mean absolute error (MAE) function, which represents the accuracy of the model to predict absolute energies of the conformers, defined below, where $\hat{y}$ represents the predicted energy and y represents the actual energy:

$$L(y - \hat{y}) = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{11}$$

The force loss refers to the negative derivative of energy loss. While derivatives are typically used during backpropagation to optimise neural networks, in this model the negative gradient is explicitly used as a target to compute the loss for force predictions. It is defined as the negative mean absolute error between predicted and true energies, given as the following:

$$L_1 = \frac{1}{N} \sum_{i=1}^{N} |y_i - \hat{y}_i| \tag{12}$$

4.1 Scalar Attribute Augmentation

In deepset and related molecular modelling architectures, atoms are represented as nodes and edges represent the interatomic interactions. The edges (nodes) are characterised by a feature given as edge weight, derived from the squared Euclidean distance between atoms.[29] The equation, for two atoms in a 3D space is given as:

$$d_{ij} = \sqrt{\sum_{k=1}^{3}(r_{i,k} - r_{j,k})^2} = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2} \tag{13}$$

In an attempt to improve the predictive capabilities of the model, various scalar transformations of the edge weight characteristics were explored to understand their influence on the model. By implementing scalar attributes, such as non-linear transformations like cubic or square root, the model gains access to different viewpoints of the same molecule, enabling it to gain a more detailed understanding of the core chemical interactions. The addition of non-linear vectors was hypothesised to increase model accuracy, since interactions such as van der Waals forces exhibit non-linear behaviour, especially given the presence of torsional effects in but-2-ene conformers.

The additional (non-linear) scalar attributes were concatenated to the edge_features tensor and passed into the d_ij_t_projection module, which computes transformed distance embeddings between nodes (atomic pairs). The example code, outlining how these attributes were implemented, was written in Figure 6 below. Edge features represent characteristics of the interaction between two atoms. By concatenating additional features, as illustrated in Figure 6, the model can gain a better understanding of spatial relationships.

```

edge_weight_sq = edge_weight ** 2
edge_weight_cube = edge_weight ** 3
edge_weight_tes = edge_weight ** 4
edge_weight_sqrt = torch.sqrt(edge_weight)
edge_weight_log = torch.log(edge_weight)
edge_weight: torch.Tensor

edge_features = torch.cat(
[
    d_ij_projection,
    edge_weight_sq.view(-1,1),
    edge_weight_cube.view(-1,1),
    edge_weight_tes.view(-1,1),
    edge_weight_sqrt.view(-1,1),
    edge_weight_log.view(-1,1),
    edge_weight.view(-1,1),
    edge_vec_flat.view(-1,1)
], dim=1
)
d_ij_t_projection = self.d_ij_transform(edge_features) # Transform distances
a_i_projection = self.a_i_transform(x[edge_index[0, :]])
```

```
    a_j_projection = self.a_j_transform(x[edge_index[1, :]])

```

Figure 6: Sample of code outlining how the additional scalar attributes were implemented into deepset.

Initially, to establish a baseline performance, the model was tested without any scalar augmentations and then with all five attributes (x^2, x^3, x^4, $\sqrt{x}$, $\log x$) afterwards. The addition of these transformations was based on the hypothesis that amplifying different ranges of atomic distances would allow the model to capture atomic interactions more effectively. One example, the addition of the square (x^2) was proposed improve capture of short range interactions by gradually amplifying at shorter distances, whilst log(x) was predicted to do the opposite, amplifying longer range interactions. Physically, since there are many different types of interactions which behave non linearly with distance, implementing this range, which all amplify different ranges was hypothesised to increase both force and energy predictive accuracy by allowing it to capture both local and long range interactions effectively.

Table 1 outlines the loss and learning rates, showing that concatenating all five scalar characteristics improved the accuracy of the energy predictions (energy loss), but caused an increase in the loss of force predictions (forces loss). This suggests that certain scalar augmentation improved the model's ability to predict the energies and forces, whilst others may hinder the modelling of its gradients, thus compromising force predictions. Table 2, below, compares the difference in loss between deepset with and without the additional five scalars concatenated.

Metric	**All Scalars**	**No Scalars**
Learning Rate	0.0002	0.0002
Test Forces Loss	0.036999	0.028074
Test Energies Loss	0.018379	0.027507
Val Forces Loss	0.038666	0.028076
Val Energies Loss	0.022945	0.01935
Avg Test/Val Forces Loss	0.0378325	0.028075
Avg Test/Val Energies Loss	0.020662	0.0234285

Table 1: Comparison of energy and force loss with and without augmented scalars (Hartree).

To develop a greater understanding and find the most accurate combination, the model was run several further times, each iteration varied as to which of the scalars were concatenated into edge features. As presented in Table 2, it became evident that certain scalars reduced the loss compared to the original model, while some, such as the logarithm, drastically increased the energy loss.

Metric	x^2 x^3 x^4 $\sqrt{x}$	x^2 x^3 x^4	x^2 x^3	x^2	x^3 x^4 $\sqrt{x}$	x^3 $\sqrt{x}$
Learning Rate	0.0002	0.0000016	0.00004	0.00004	0.001	0.0002
Test Forces Loss	0.027261	0.14131	0.02582	0.025993	0.030369	0.026497
Test Energies Loss	0.011171	0.14588	0.01571	0.018682	0.037639	0.015335
Val Forces Loss	0.027695	0.14865	0.025757	0.025897	0.030816	0.026223
Val Energies Loss	0.016746	0.14112	0.015148	0.017632	0.044079	0.013202
Avg Test/Val Forces Loss	0.027478	0.14498	0.0257885	0.025945	0.0305925	0.02636
Avg Test/Val Energies Loss	0.0139585	0.1435	0.015429	0.018157	0.040859	0.0142685

Table 2: Comparison of energy and force loss with different combinations of non-linear features (Hartree).

The incorporation of the tesseract (x^4)and logarithmic ($\log x$) transformations negatively impacted the accuracy of the model when predicting both forces and energies. The tesseract transformation introduced extreme non-linearity, which overly amplified minor variations in atomic distances. This likely degraded model stability by producing distorted feature representations.

Much like the tesseract, the logarithmic function compresses differences at larger values (longer interatomic distances) as opposed to close values, while amplifying variations at shorter distances, which hinders the model's ability to calculate 1,3-diaxial interactions and electrostatic repulsions in the but-2-ene conformers. From a physical perspective, long range interactions contain the most critical information, consisting of strong interatomic forces, such as the hyperconjugation and Van der Waals forces observed in but-2-ene. By disproportionately amplifying these values, the model's ability to effectively distinguish between subtle variations in the energies between conformers can be compromised, resulting in the observed loss of precision.

The combination of cubic (x^3) and square root ($\sqrt{x}$) transformations yielded the highest accuracy energy predictions, as outlined in Table 2. This specific pairing offers complementary sensitivity across different interaction distances. The cubic transformation improves the model's ability to capture short-range atomic interactions by proportionately amplifying the influence of nearby atoms, which is crucial for understanding the atomic structure. In contrast, the square root transformation improves representation of longer range interactions through a more flattened distance response, which somewhat relates to the gradual decay of electrostatic effects observed throughout molecular modelling with distance. However, this pairing yields no increase in predicting forces, which was systematically greater than predictions of energies. Without square (x^2) features, the model misses certain interaction patterns that are particularly important for force predictions in but-2-ene, this is reinforced by the run with square alone, where the force prediction outperforms the deepset with cubic and square root, although only slightly.

Figure 7 shows how the energy loss varies across the training of the model and Figure 8 compares the force loss throughout training, the x-axis: Global Step, represents the number of optimisation steps. Throughout the run, the disparity in loss between the Cub-Sqrt scalars against the other two is evident, cub-sqrt begins at significantly lower loss, and remains the lowest throughout, even showing resistance to the spike at 1000, peaking to only 0.08 hartree, which is negligible in comparison to the model without scalars, which peaked at 10.5 hartree. Suggesting that the training process is more stable, therefore is also more computationally efficient, fewer epochs are required for the model to converge, as well as typically providing greater accuracy.

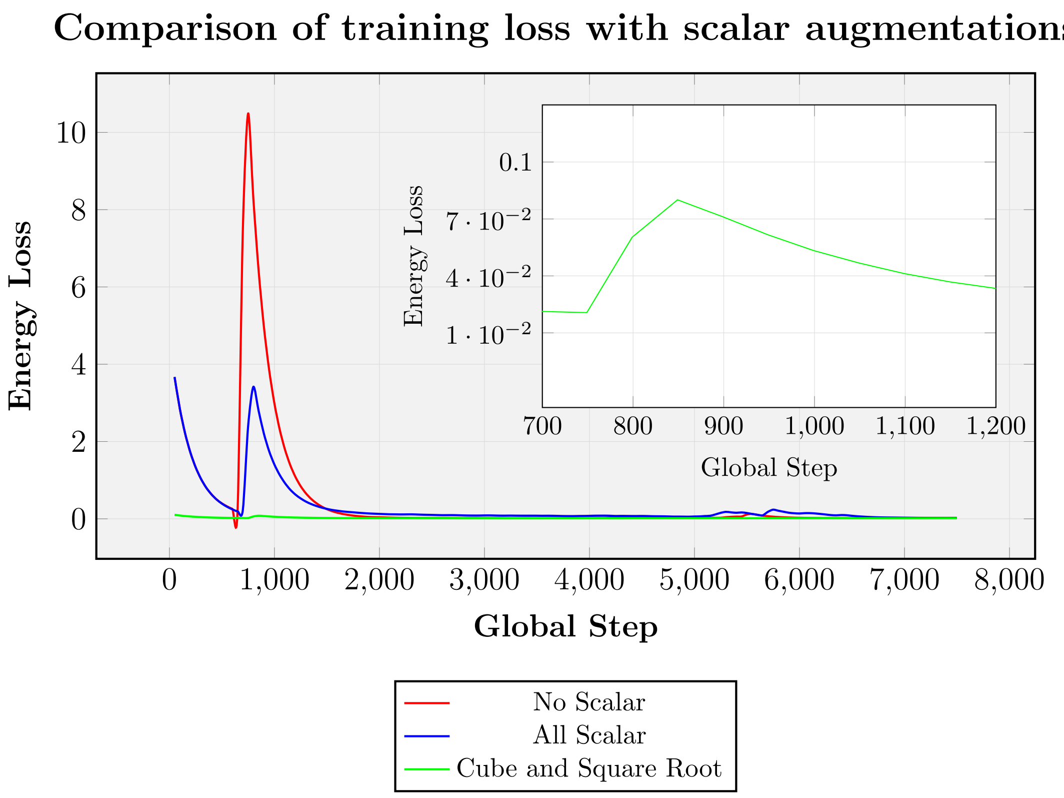


Figure 7: Graph comparing the energy loss throughout training on but-2-ene for deepset without any additional scalars, with all additional scalars, and with only cube (x^3) and square root ($\sqrt{x}$) added. The inset highlights a jump in the Cube and Square Root curve between 700 and 1200 global steps.

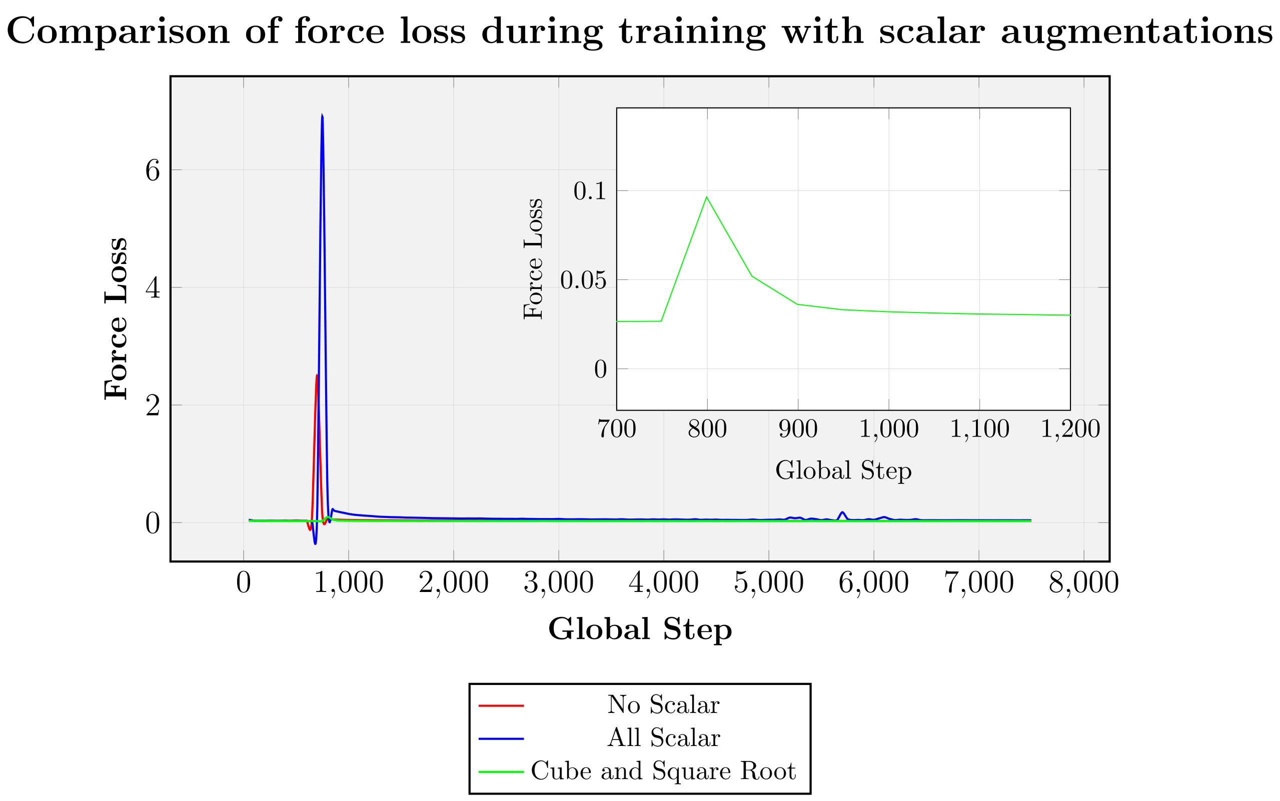


Figure 8: Graph comparing the force loss throughout training on but-2-ene for deepset without any additional scalars, with all additional scalars, and with only cube (x^3) and square root ($\sqrt{x}$) added. The inset highlights a jump in the Cube and Square Root curve between 700 and 1200 global steps.

In the context of but-2-ene conformers, this complementary non-linear representation allows the model to account for both short-range interactions, such as the hyperconjugation between the methyl group and the carbon double bond, and longer-range 1,3-diaxial interactions. These features are especially relevant when calculating potential energy surfaces (PES) due to their large contribution to molecular energy. Compared to the standard linear embedding approach, which was shown in Figure 7 to be less accurate, this non-linear combination offers a more chemically relevant representation of interatomic interactions. Notably, this performance increase is achieved without the addition of numerical instabilities that are commonly associated with higher-order polynomials (such as tesseract), which often result in amplification or oversmoothing of interatomic distances, regardless of the scale.

The significant improvement suggests that incorporating non-linearities into edge representations offers a more thorough approach for the model to calculate the potential energies and forces, as opposed to relying on multiple message-passing layers to implicitly learn these relationships. Additionally, requiring a low increase in computational demand, increasing the model's run time by just 2 minutes on average.

4.2 Integration of Graph Neural Network (GNN) Architecture

The purpose for implementing different GNNs is to explore how different message passing schemes and aggregation mechanisms affect the model's ability to capture complex molecular interactions. GNNs offer diverse approaches to information propagation across atomic systems, specifically the message-passing paradigm, which could potentially capture the electron density redistribution during conformational changes, in this case for but-2-ene. Previous computational approaches to conformational analysis, such as those by Xu *et al.* (2023) and Wang et al. (2022), have demonstrated the challenges in modelling rotational energy barriers in alkenes, but have not fully leveraged GNN architectures within their respective models.[30;31]

The model was tested with various types of convolutional layers and different numbers of layers within the neural network to understand their impact on accurately predicting the energies and forces of conformations.

4.2.1 SAGEConv

SAGEConv was chosen due to its neighbourhood sampling and aggregation approach to message passing, which is largely relevant to but-2-ene, which only has four carbon atoms present. By combining central atom features with those of neighbouring atoms, it was hypothesised that the model's ability to predict energies would increase upon addition, however, as portrayed in Table 3, this was not the case. The example code, introducing SAGE into the model was shown as below in Figure 9.

```python

        # Node embeddings updated by SAGE
        self.gnn = SAGEConv(in_channels, out_channels, aggr = 'mean')

        # Pass through SiLU
        gnn_output = F.silu(self.gnn(x, edge_index))
        # Concat updated features from source and target nodes
        gnn_edge_features = torch.cat([gnn_output[edge_index[0]],
        gnn_output[edge_index[1]]], dim=1)

        # Concat projected distance features and transform
        gamma_projection = self.gamme_transform(torch.cat([a_i_projection,
            a_j_projection, d_ij_t_projection, gnn_edge_features], dim=1))

```

Figure 9: Sample of code outlining how SAGE was implemented into deepset.

From Table 3, it is evident that the addition of the SAGE layer on top of the deepset reduced the model's

Metric	**SAGE Sum**	**SAGE Max**	**SAGE Mean**	**SAGE 3 Layer**
Learning Rate	0.0000016	0.001	0.001	0.0000016
Test neg-Forces l1 loss	0.027912	0.027622	0.027912	0.028093
Test Energies loss	0.017568	0.017212	0.020647	0.020544
Val neg-Forces l1 loss	0.03751	0.02782	0.028047	1.73053
Val Energies loss	0.050442	0.020743	0.020319	6.82534
Avg Test/Val Forces l1 loss	0.03271	0.02772	0.02798	0.87931
Avg Test/Val Energies loss	0.03400	0.01898	0.02048	3.42294

Table 3: Comparison of different model configurations and their energy and force losses (Hartree).

ability to predict both energies and forces on but-2-ene conformers. Most notably, increasing the number of GNN layers from one to three resulted in an over 100-fold decrease in accuracy. While this outcome was initially unexpected, it can be attributed to oversmoothing, where deeper GNNs cause node representations to become increasingly similar. Since but-2-ene is a small molecule, adding so many layers can affect local specificity, making some coordinates indistinguishable with each message pass. To avoid this, the model was restricted to one layer for the remainder, where it performed optimally, showing that addition of GNN on top of deepset needs to be extremely careful with how information is passed and transformed.

One benefit of SAGE is that the aggregation type can be changed. The aggregation type defines how the messages are combined before being passed through the layer.

By switching from mean to max, the accuracy of the model when predicting energies improved, albeit slightly. However, this affirms that the addition of SAGE GNN does not improve accuracy. The mean aggregation type smooths over neighbours, by calculating the average value, whereas the max type selects the most prominent value, which, although it is typically unstable during training, has been shown to have a performance edge in real tasks in prior studies, such as one by Schweidtmann *et al.*, which investigated all three aggregation types on a range of datasets.[32] The sum aggregation method, which simply adds neighbouring features without normalisation, was hypothesised to perform poorly. This is supported by prior knowledge in molecular modelling, where sum-based aggregation can produce inconsistent representations, commonly when atoms have varying numbers of bonded neighbours. This results in a diminished performance compared to normalised strategies such as mean or max, as was shown in Table 3.

4.2.2 EDGEConv

Although the implementation of a GNN with SAGE layer did not improve the model's predictions of forces or energies, different convolutional layers must be tested, since each behaves and extracts features differently. EDGE focuses on capturing edge attributes (bond distances and angles), which was hypothesised to assist the model in predicting conformations, thus increasing the accuracy of energy predictions. The downside however, is that deepset already focuses on this with edge_features and with the addition of prior scalars (introduced in Chapter 3.1), it was uncertain whether EDGE would be advantageous to the model in this exact application. EDGE was implemented with one layer and was trained for 150 epochs, with the results outlined below, in Table 4. The sample code, outlining how EDGE was implemented into the model is given below as, Figure 4. EDGE was applied to learn representations using max aggregation, and was subsuequently refined via a two layer transformation with SiLU non linearity.

```

    # Passing through EDGE convolutional layer with 'max' aggregation type
    self.edge = EdgeConv(256, hidden_channels, aggr='max')
                    nn_edge = Sequential(
                            nn.Linear(512, hidden_channels),
                            SiLU(),
                            nn.Linear(hidden_channels, hidden_channels)
                    )

```

Figure 10: Sample of code outlining how EDGE was implemented into deepset.

Metric	**EDGE Scalar**	**SAGE Max**	**Cub-Sqrt**
Learning Rate	0.0002	0.001	0.0002
Test Forces L1 Loss	1.45016	0.027622	0.026497
Test Energies Loss	3.85752	0.017212	0.015335
Val Forces L1 Loss	1.45092	0.02782	0.026223
Val Energies Loss	12.7303	0.020743	0.013202
Avg Test/Val Forces Loss	1.45054	0.027721	0.02636
Avg Test/Val Energies Loss	8.29391	0.018978	0.014269

Table 4: Comparison of EDGE, SAGE and the prior version without GNN (Hartree).

The energy and force loss were drastically increased upon the replacement of SAGE with EDGE. This change is likely due to the calculation of the conformational features, since EDGE focuses on edge features in an attempt to enhance the representation of geometric features.

This may have interfered with the prior additions of scalar attributes, which were introduced to enhance the representation of edge features. Since the GNN was applied following the concatenation of these scalar attributes, it is plausible that EDGE subsequently overwrote the result by injecting its own edge features, leading to a mismatch from the original embedding. Given only one layer was added, it is unlikely that the result is due to oversmoothing, but the model could potentially double counting, with two competing

Metric	EDGE No Scalar	EDGE Scalar
Learning Rate	0.0000016	0.0002
Test Forces Loss	0.027732	1.45016
Test Energies Loss	0.020587	3.85752
Val Forces Loss	0.030082	1.45092
Val Energies Loss	0.034315	12.7303
Avg Test/Val Forces Loss	0.028907	1.45054
Avg Test/Val Energies Loss	0.027451	8.29391

Table 5: Comparison of energy and force loss for EDGE, with and without augmented scalars (Hartree).

pathways for the same information conflicting within the calculations, a concept previously studied by Liu *et al* whom showed that identical operations can have opposite outcomes dependent on the execution.[33] This hypothesis was tested by the model being run with one EDGE GNN layer, and the scalar attributes concatenated to edge_features were removed to see if the accuracy improves, and thus understand the reasoning behind the data previously observed. The results, outlined in Table 5, are shown below:

From the results, it is evident that the hypothesis proposed for the disproportionate loss of accuracy was correct. By removing the previously appended scalar attributes and allowing EDGE to pass its own features, the calculated loss returned to an expected value. Despite this, EDGE still performed slightly worse than SAGE, which can be attributed to how each layer participates in message passing compared to the pre-existing architecture of TorchMD, with SAGE's neighbourhood sampling approach better suited in conjunction with the augmented edge features.

4.2.3 GATConv

In contrast to SAGE and EDGE, GAT is one of the most widely cited conformational layers of the PyTorch geometric package (pyg). Introduced by Velickovic *et al* (2017), the neighbourhood aggregation and multi-head attention approach is highly regarded in computational chemistry due to allowing the model to focus on the closest and strongest interatomic interactions such as hyperconjugation, with different heads tasked to specialise in different types of interaction.[34] The implementation of GAT into deepset is given below, as Figure 11. The dropout is a regularisation technique which is applied to the attention coefficients by GAT during the message passing, it prevents the model from becoming too reliant on specific neighbours and is intended to encourage generalisation.

```

    self.gat = GATConv(
                        in_channels=128, # Input features per node
                        out_channels=hidden_channels,
                        heads=4, # Attention heads
                        concat=True, # Concat output from all heads
                        dropout=0.1
              )

```

Figure 11: Sample of code outlining how GAT was implemented into deepset.

Given the nature of GAT and the TorchMD architecture, it was expected to perform well, although it was still uncertain whether GNN would increase the accuracy, since the previous two variations did not. The results, in Table 6, comparing GAT with and without the additional scalars, are shown below.

Metric	**GAT No Scalar**	**GAT with x^3,$\sqrt{x}$**	x^3,$\sqrt{x}$
Learning Rate	0.0000016	0.00004	0.0002
Test Forces Loss	0.02787	0.027989	0.026497
Test Energies Loss	0.023051	0.016086	0.015335
Val Forces Loss	0.060313	0.028056	0.026223
Val Energies loss	0.37685	0.017516	0.013202
Avg Test/Val Forces_l1_loss	0.0440915	0.028023	0.02636
Avg Test/Val Energies_loss	0.1999505	0.016801	0.0142685

Table 6: Comparison of energy and force loss for GAT with and without augmented scalars (Hartree).

Unlike the EDGE model, GAT exhibited greater accuracy when paired with concatenated edge features, as introduced in Chapter 4.1. The results showed that GAT improved the energy predictions relative to both SAGE and EDGE. However, the force predictions remained similar across all variants, with no significant improvement observed.

One possible explanation is that adding the GNN on top of deepset improves energy predictions by enhancing the modelling of global interactions, but does not significantly affect the local gradients from which forces are calculated. Since forces are calculated as the negative gradient of energy with respect to atomic positions, the model must be able to capture fine spatial details accurately. Additionally, the radial basis function (RBF), which was already incorporated to model spatial relationships, is expected to play a key role in force prediction with its contribution to the distance filter, restricting the impact of the different convolutional architectures within the model.

The energy and force prediction loss during training for each GNN variation is shown in Figures 12 and 13 below. Due to using SGD with the Adam optimiser, force loss training was expected to be more unstable since the optimisation can overshoot and is known to cause small instabilities near convergence. However, it is still evident that the EDGE training was drastically more unstable than GAT and SAGE, which is reflected both in the loss curves and final performance tables, where it dramatically underperformed.

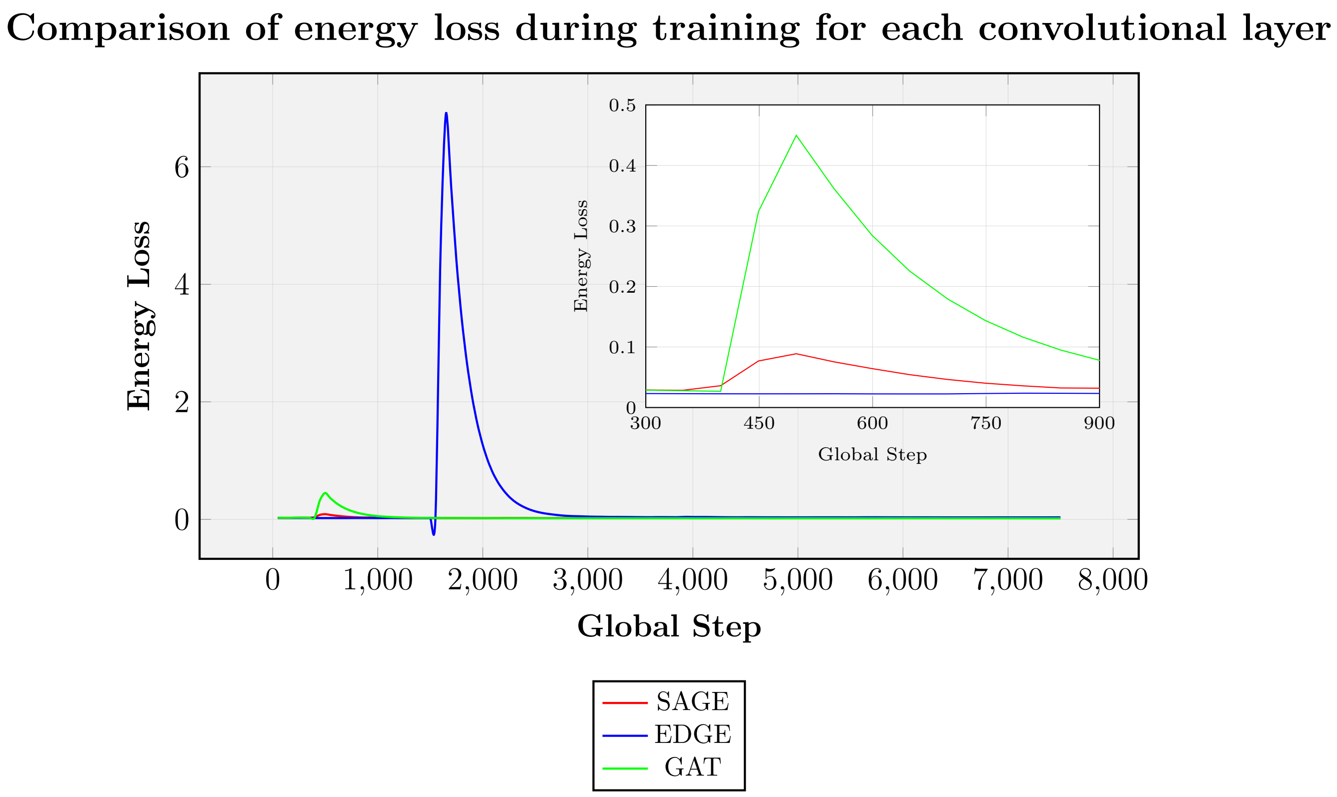


Figure 12: Graph comparing the energy loss throughout training on but-2-ene for deepset with the greatest performing variant of each convolutional layer. The inset highlights the region between 300 and 900 global steps to show finer trends across models.

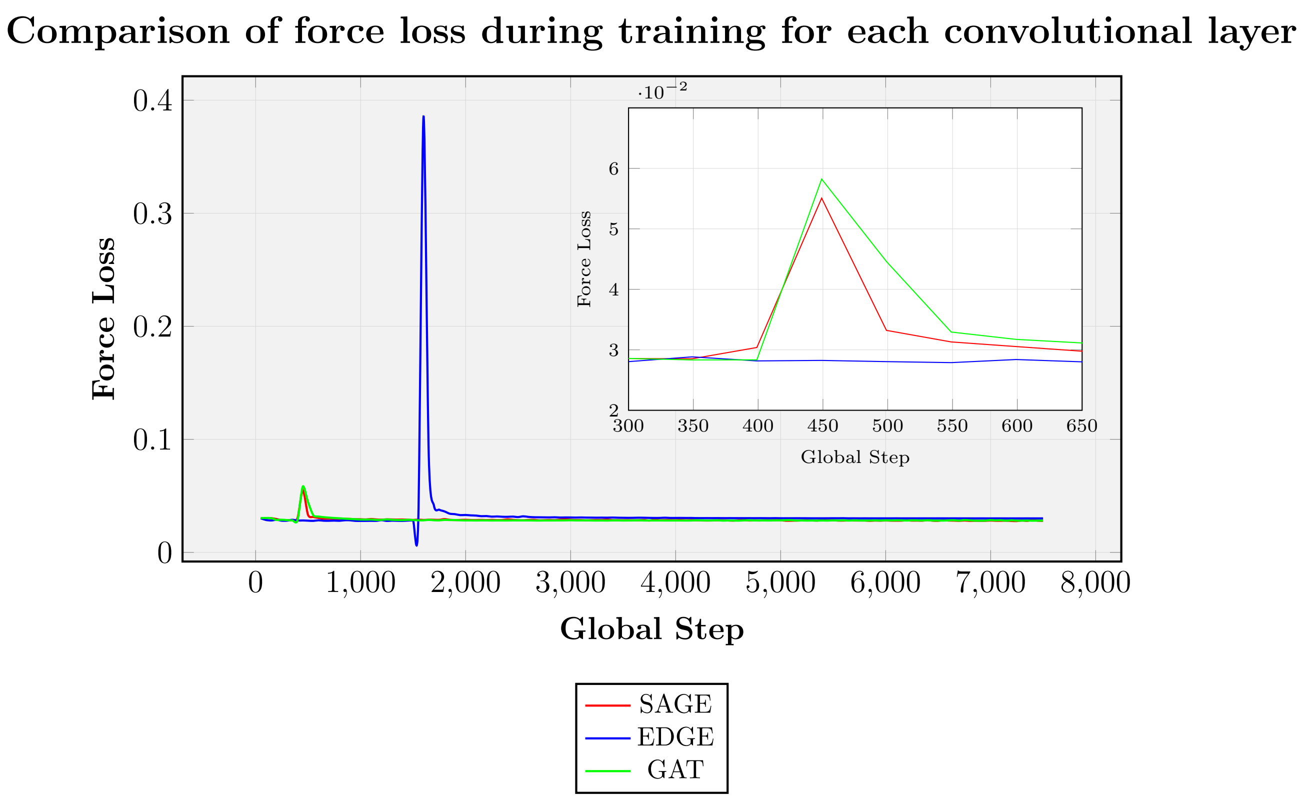


Figure 13: Graph comparing the force loss throughout training on but-2-ene for deepset with the greatest performing variant of each convolutional layer. The inset highlights the region between 300 and 900 global steps, where larger fluctuations are observed.

In contrast, the energy predictions demonstrated a clear improvement with the use of GAT, although they still slightly underperformed compared to deepset without any GNN augmentation. Notably, GAT achieved a 63% increase in energy prediction accuracy relative to EDGE, which was consistent with expectations based on previously discussed studies. The attention mechanism in GAT contributed to this improvement by selectively emphasising the seemingly most important atomic interactions towards the system's total energy, a superior technique compared to the neighbourhood sampling method of SAGE also. This allowed the model to capture subtle energetic differences between conformers more effectively and to respond better to small structural variations in the but-2-ene dataset.

4.3 Enhanced Radial Basis Function (RBF) via Scalar Transformations

RBF is used in machine learning to measure the distance between points in a multidimensional space, specifically transforming interatomic distances into feature vectors that capture how atomic influences decay with separation, preserving rotational and translational invariance, defined in TorchMD as:[15]

$$e_k^{\mathrm{RBF}}(d_{ij}) = \phi(d_{ij}) \exp\left(-\beta_k \left(\exp(-d_{ij}) - \mu_k\right)^2\right) \tag{14}$$

The RBF function contributes to the distance filter, which is multiplied by the neighbourhood embedding to generate the final atom embedding. By improving the RBF's capacity to capture detailed information about interatomic distances, the quality of the atom embeddings is enhanced, which can help reduce overall model loss. In this study, the RBF was modified by introducing additional scalar attributes into the distance_expansion RBF layer, as shown below as :

```

        # Applies RBF expansion to non linear transformations
        edge_attr_dist = self.distance_expansion(edge_weight)
        edge_attr_sq = self.distance_expansion(edge_weight_sq)
        edge_attr_cub = self.distance_expansion(edge_weight_cub)
        edge_attr_tes = self.distance_expansion(edge_weight_tes)
        edge_attr_sqrt = self.distance_expansion(edge_weight_sqrt)
        edge_attr_log = self.distance_expansion(edge_weight_log)

    # Concat chosen transformed RBF embeddings
    edge_attr = torch.cat([edge_attr_dist, edge_attr_cub, edge_attr_sqrt],
            dim=1)

    # Computes cutoff values and applies them to the projected edge features
    C = self.cutoff(edge_weight)
    d_ij_projection = self.distance_proj(edge_attr) * C.view(-1, 1)

```

Figure 14: Sample code outlining how the additional RBF attributes were added to deepset.

The model was trained again, without any GNN, instead using the proven scalar features (cubic and square root) found to increase the accuracy. The results shown in Table 7 are given below.

Metric	**RBF** x^3 $\sqrt{x}$	**RBF** x^2 x^3	**RBF** x^2 x^4 $\ln x$	**RBF** x^2 x^4	**No RBF**
Learning Rate	0.0000016	0.000008	0.0000016	0.000008	0.0002
Test Forces Loss	0.027891	0.027829	0.027971	0.027522	0.026497
Test Energies Loss	0.019049	0.018638	0.018367	0.018486	0.015335
Val Forces Loss	0.031474	0.028688	0.038438	0.030376	0.026223
Val Energies Loss	0.026550	0.021831	0.092588	0.023074	0.013202
Avg Test/Val Forces Loss	0.029683	0.028259	0.033205	0.028949	0.026360
Avg Test/Val Energies Loss	0.022799	0.020235	0.055478	0.020780	0.014269

Table 7: Comparison of energy and force loss with and without modified RBF (Hartree).

The results indicate that the incorporation of non-linear edge attributes to the RBF was not beneficial to the model, despite a variety of combinations being tested. The most accurate combination was found by appending square and cubic non-linear transformations to edge_attributes before passing the RBF function. In contrast to edge_features where the cubic and square root combination was best, edge_attributes did not perform well with this pairing, showing that there was no significant correlation between these two.

The expansion of the feature space by appending additional attributes with new, seemingly redundant variables leads to pronounced overfitting of the RBF.[35] Overfitting derives from the addition of too many features, more common if they aren't informative to the model (redundant), evident from the variation with square, tesseract, and logarithm appended, which has the most features (three additional) and exhibited staggeringly worse energy and force prediction accuracy.[36]In addition to overfitting, such additions to the RBF have been shown to impair generalisation to unseen geometries (data), most commonly when these additions do not correspond to meaningful variations, with it observed that the number of features exponentially grows the amount of data needed to achieve reliable generalisation.[37]

Importantly, the prediction of energies was impacted more than the prediction of forces from this alteration. Given that the energy loss is calculated by MAE, and force loss is the negative derivative of energy loss, which is a local derivative and thus only sensitive to local gradients, the disparity can be explained. The MAE aggregates deviations across the global energy landscape and is sensitive to any global distortion, whilst the force calculations, being a local vector, are less sensitive.

4.4 Analysing Butene Dataset as a Time Series

The model chosen for evaluation was the configuration found to be most accurate throughout but-2-ene training, specifically the model employing additional cubic and square root edge features without any additional GNN layers or enhanced RBF attributes. To gain an understanding of the model performance from another perspective, a time series plot was created by loading this deepset version from a checkpoint, importing the data and using plotly to generate the graph shown in Figure 15.

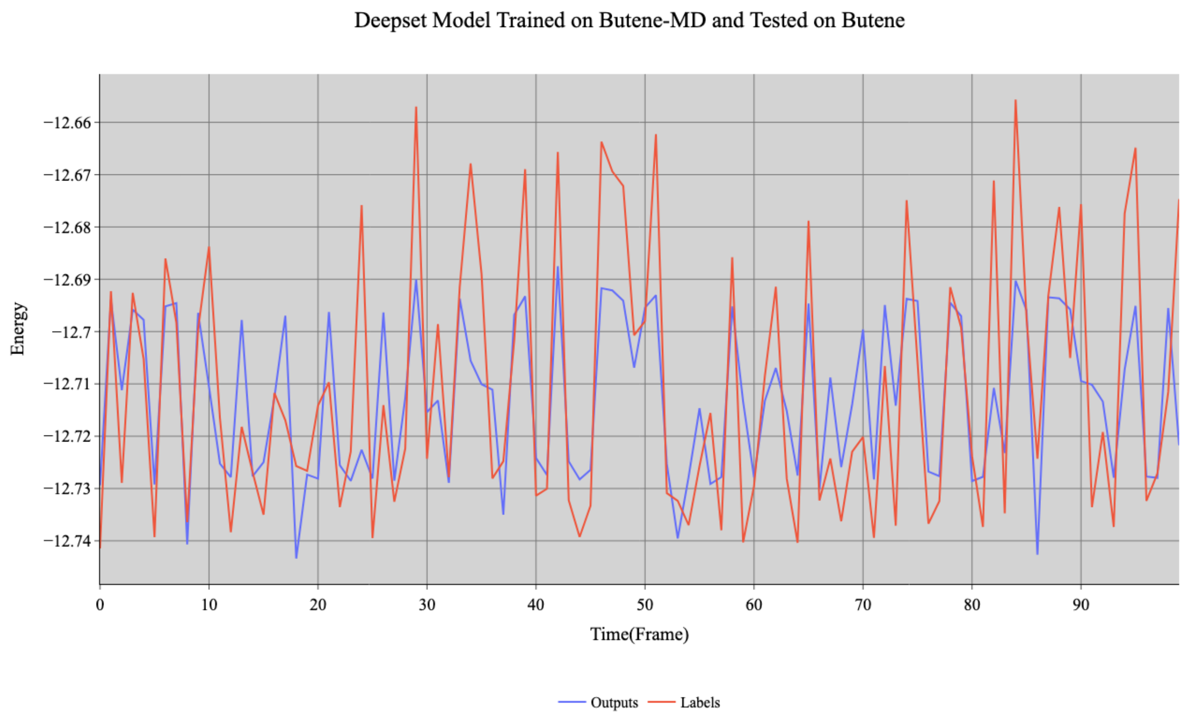


Figure 15: Time series plot for deepset model with cubic and square root scalar augmentations tested and trained on but-2-ene.

From the time series plot, it is evident where the model lacks predictive accuracy. The red curve represents the reference energies of the but-2-ene conformers, whilst the blue curve shows the deepset model predictions. The greatest discrepancy is observed at the lowest-energy conformers (trans-but-2-ene), where the model consistently overestimates the energies throughout the plot. Since the cutoff radius was set to 10 Å, the model captures all relevant bonded and non-bonded interactions, ruling out the possibility that the errors derived from insufficient atomic representation.

Instead, the lack of accuracy when predicting the energy of conformers with highest energies can be attributed to oversmoothing, a known issue in neural networks that employ permutation-invariant aggregation mechanisms such as deepset.[38] In this architecture, atomic features are aggregated across neighbours using a scatter-based sum operation (shown below), which ensures that the model's output is invariant to the order of the atoms. While permutation invariance is crucial for molecular systems, where atom order should not affect properties, the sum-based aggregation treats all neighbouring atoms equally, regardless of their specific chemical environments and cutoff. As a result, the model averages over local features, leading to the loss of more subtle structural information, which is crucial due to the small differences in atomic distances between the but-2-ene conformers.

```

        # Combine outputs from multiple experts at the edge level by summation
        edge_level_output = torch.sum(torch.stack(experts_contributions, dim=0), dim=0)

        # Aggregate outputs by summing all edges
    atom_level_output_x = scatter(edge_level_output, edge_index[0], dim=0,
            reduce="sum")

```

Despite the cis and trans conformers of but-2-ene differing significantly in energy, their interatomic distances only subtly vary (shown in Figure 16). The model focuses on encoding interatomic distances and is therefore unable to perfectly preserve these subtle local differences after layers of aggregation. As a result, there is a systematic underestimation of the energy at extremities (trans conformer). This behaviour reflects a fundamental limitation of simple permutation-invariant networks like deepset when modelling small but chemically significant energetic differences. However, invariance is crucial in molecular modelling, so it is important to consider alternate approaches which better tackle this issue.

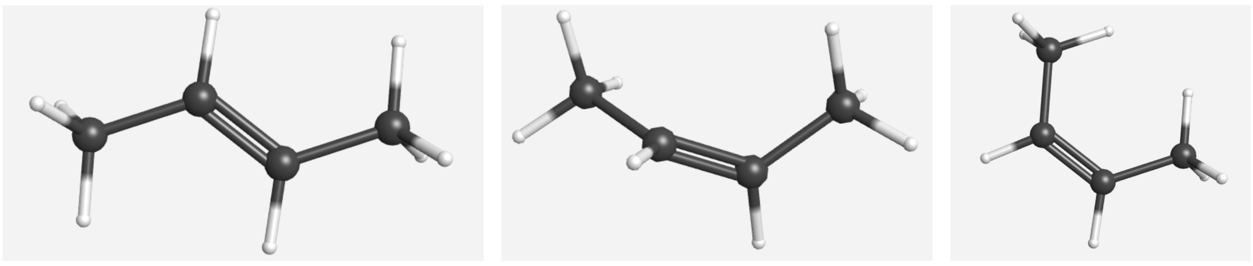


Figure 16: Comparison of Cis, Halfway, and Trans Conformations of But-2-ene.

4.5 Generalisation of Model Trained on Butene over Pentene

To further understand the performance of deepset, it was tested on an unseen pent-2-ene dataset, with all parameters kept the same as during training. Much like but-2-ene, the pent-2-ene dataset contains a series of conformers and their respective energies, and deepset was tasked with predicting these energies and forces for each unseen pent-2-ene conformer.

Prior research, such as the study by Ong *et al.* (2023), tested state-of-the-art models for predicting binding affinities and found that each exhibited limited generalisation.[39] Given this, it is evident that many models struggle to extrapolate beyond the chemical environment they are trained on. Despite being tested in a different application, this observation suggests that deepset may not be learning fundamental principles of molecular interactions. Instead, interpolating directly within the training data, leading to a sharp decline in performance when applied to unseen molecules with even modest structural differences, such as pent-2-ene.

This reflects a broader challenge in molecular modelling using machine learning, the need to move beyond dataset specific fitting and towards effective generalisation, which remains a key barrier to reliable, predictive chemistry. The model, using the same deepset version with cubic and square root added was loaded and tested on pent-2-ene to offer clear insight into these strengths and limitations, helping to evaluate whether deepset can function as a reliable predictor for unseen molecules. The results are presented in the time-series chart in Figure 17.

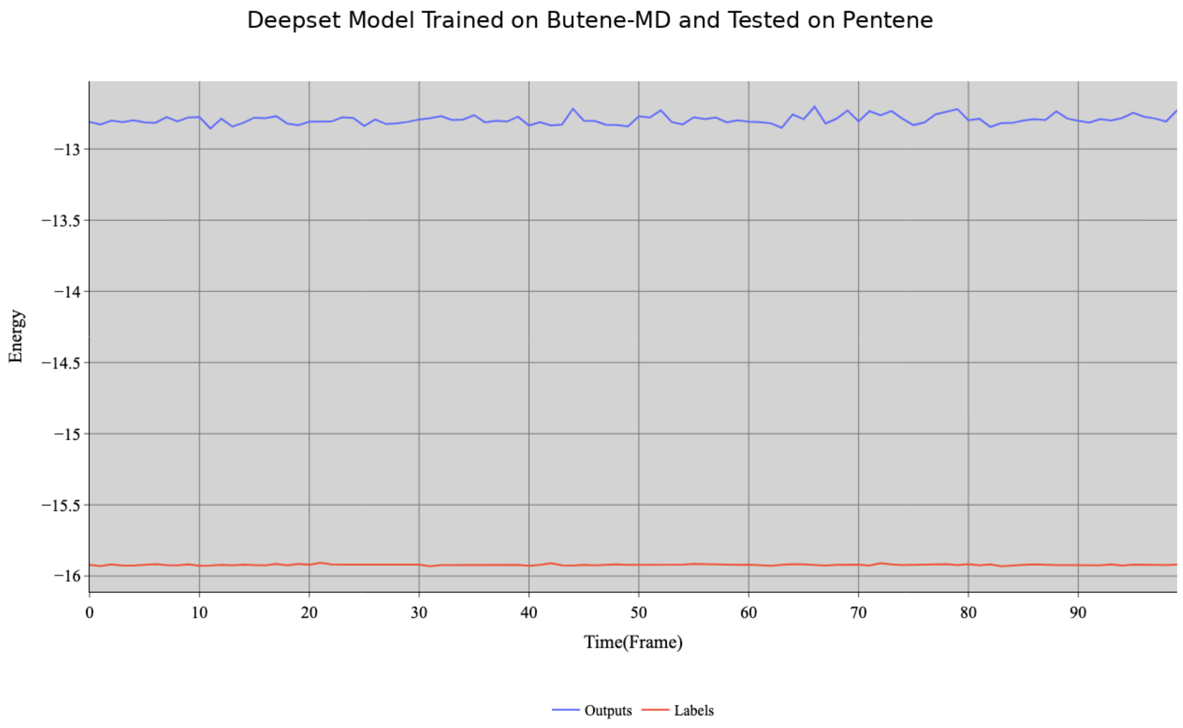
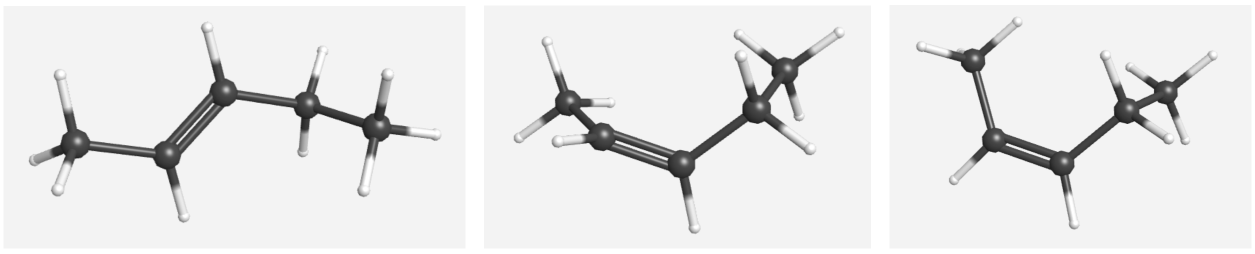


Figure 17: Time series plot for deepset model with cubic and square root scalar augmentations tested on pent-2-ene.

From the time series plot, it is evident that deepset lacks the ability to generalise accurately on unseen data. By training exclusively on but-2-ene, the model was able to learn and consistently predict the forces and energies of but-2-ene conformers to a high degree of accuracy. However, when applied to an unseen example of pent-2-ene, a structurally similar alkene, the predictions (blue) differed significantly from the actual values (red), executing with accuracy over 100 times worse. The time series plot for but-2-ene peaked at a disparity of 0.03 hartree, whilst the pent-2-ene plot predicted values were consistently too low by about 3.00 hartree, remaining around the expected energies for but-2-ene predictions instead, thus deepset struggles to adapt to the greater energy of pent-2-ene and is over reliant on the training, proving it is an interpolative not an extrapolative model.

Given this, it is also likely the model was overfitting to the training data, one reason for this could be the number of train vs test epochs used, although unlikely. Three conformers of pent-2-ene are given below in Figure 18, showing their similarity to the but-2-ene conformers previously given.



Trans-Pentene Halfway Conformation Cis-Pentene

Figure 18: Comparison of Cis, Halfway, and Trans conformations of Pent-2-ene.

To gain a better understanding of how deepset predicts these energies, the model was updated to compute the predicted energies for each atom in the conformer, the added code is shown below as Figure 19, from this it became clear how the total energies were calculated, since the energies of atoms should not change substantially through the different steps (conformations). The atom_types corresponds to the molecules in but-2-ene, which were altered when using for pent-2-ene

```python

        # Define atom types for but-2-ene
        atom_types = ["C", "C", "C", "C",
                      "H", "H", "H", "H", "H", "H", "H", "H"]

        # Initial time and time increment per step
        initial_time = 0.0
        time_interval = 5.0

        # Loop over all prediction steps
        for i in range(n_steps):
            # Get predicted energies and compute total energy
            energies = predicted_energies[i]
            total_energy = energies.sum()
            time = initial_time + i

            # Log the energies in an external csv file
            for idx, (atom_type, atom_energy) in enumerate(zip(atom_types, energies)):
                csv_data.append({
                    "step": i,
                    "time_fs": time,
                    "atom_idx": idx,
                    "atom_type": atom_type,
                    "atom_energy": atom_energy.item(),
                    "total_energy": total_energy.item()
                })

```

Figure 19: Sample of code outlining how the per-atom energies were generated.

From a chemical perspective, the energy per atom is expected to be the same for different conformers, varying only slightly as the bond angles vary. The energy per atom for but-2-ene conformers is given below as Table 8, the step corresponds to the time (frame) on the x-axis of the time series plot in Figure 15.

Step	Carbon Atoms				Hydrogen Atoms							
	C1	C2	C3	C4	H1	H2	H3	H4	H5	H6	H7	H8
0	-0.073	-1.473	0.532	-0.096	0.356	0.409	1.134	-0.779	-1.260	1.026	-0.417	-0.318
5	1.273	-1.284	-0.185	0.614	-0.341	-0.300	1.723	-0.823	0.569	0.815	-0.390	-0.192
10	1.263	-2.194	-0.270	0.851	-0.045	-0.138	3.043	-1.252	0.448	1.001	-0.374	-0.265
15	1.273	-1.225	-0.177	0.598	-0.360	-0.306	1.637	-0.790	0.587	0.809	-0.388	-0.183
20	1.131	-2.518	-0.292	0.923	0.133	-0.049	3.507	-1.382	0.356	1.014	-0.367	-0.243
25	-0.073	-1.458	0.534	-0.098	0.354	0.404	1.109	-0.769	-1.254	1.020	-0.417	-0.315
30	1.261	-2.281	-0.281	0.876	-0.018	-0.130	3.169	-1.292	0.432	1.016	-0.377	-0.271
35	1.260	-2.202	-0.271	0.853	-0.045	-0.146	3.052	-1.251	0.444	1.000	-0.377	-0.263
40	-0.087	-2.289	0.434	0.112	0.632	0.540	2.318	-1.169	-1.383	1.178	-0.415	-0.385
45	1.268	-1.955	-0.244	0.790	-0.122	-0.173	2.701	-1.141	0.480	0.957	-0.380	-0.246
50	-0.095	-2.556	0.407	0.180	0.723	0.576	2.699	-1.286	-1.415	1.231	-0.410	-0.405
55	1.133	-2.109	-0.251	0.811	-0.011	-0.113	2.919	-1.192	0.409	0.933	-0.371	-0.211
60	-0.072	-1.462	0.532	-0.097	0.353	0.406	1.118	-0.772	-1.256	1.022	-0.417	-0.316
65	1.275	-1.355	-0.193	0.634	-0.316	-0.287	1.826	-0.855	0.564	0.828	-0.388	-0.196
70	1.259	-2.487	-0.303	0.931	0.050	-0.097	3.464	-1.387	0.405	1.053	-0.375	-0.286
75	1.280	-1.339	-0.187	0.632	-0.319	-0.285	1.811	-0.851	0.561	0.827	-0.393	-0.196
80	1.265	-2.163	-0.262	0.843	-0.054	-0.142	3.004	-1.236	0.450	0.999	-0.376	-0.263
85	-0.069	-1.500	0.528	-0.085	0.368	0.413	1.173	-0.790	-1.260	1.027	-0.418	-0.317
90	1.264	-2.295	-0.286	0.882	-0.012	-0.126	3.194	-1.303	0.426	1.020	-0.379	-0.274

Table 8: Per-Atom Energies of But-2-ene at Different Timesteps, in Hartree)

Table 8 presents the partial energy prediction performance of deepset on the but-2-ene dataset. It is noteworthy that this comparison is fully unsupervised, devoid of any ground truth labels. Consequently, the primary objective of this comparison revolves around assessing the similarity in energy levels of each atom within distinct test samples. While the molecules in the but-2-ene dataset exhibit a wide range of conformations, the molecule being evaluated has extensive experience with various permutations, implying that multiple domains of energy changes are anticipated for each atom and the ordering is not meaningful.

From Table 9, it becomes apparent where the model exhibits limitations in its predictive capabilities regarding distribution shift. When pent-2-ene conformers with the same level of theory as the but-2-ene dataset were utilised to evaluate the generalisability of the aforementioned model, it was evident. Figure 17 illustrates the overall performance of the model, while Table 9 elucidates how the total energy per molecule is being formed.

Step	Carbon Atoms					Hydrogen Atoms									
	C1	C2	C3	C4	C5	H1	H2	H3	H4	H5	H6	H7	H8	H9	H10
0	-0.073	-1.473	0.532	-0.096	0.356	0.409	1.134	-0.779	-1.260	1.026	-0.417	-0.318	-0.887	1.040	-0.417
5	1.273	-1.284	-0.185	0.614	-0.341	-0.300	1.723	-0.823	0.569	0.815	-0.390	-0.192	-2.369	0.895	-0.781
10	1.263	-2.194	-0.270	0.851	-0.045	-0.138	3.043	-1.252	0.448	1.001	-0.374	-0.265	-3.171	1.675	-1.327
15	1.273	-1.225	-0.177	0.598	-0.360	-0.306	1.637	-0.790	0.587	0.809	-0.388	-0.183	-2.316	0.850	-0.742
20	1.131	-2.518	-0.292	0.923	0.133	-0.049	3.507	-1.382	0.356	1.014	-0.367	-0.243	-3.316	1.935	-1.519
25	-0.073	-1.458	0.534	-0.098	0.354	0.404	1.109	-0.769	-1.254	1.020	-0.417	-0.315	-0.872	1.030	-0.407
30	1.261	-2.281	-0.281	0.876	-0.018	-0.130	3.169	-1.292	0.432	1.016	-0.377	-0.271	-3.242	1.746	-1.384
35	1.260	-2.202	-0.271	0.853	-0.045	-0.146	3.052	-1.251	0.444	1.000	-0.377	-0.263	-3.166	1.676	-1.335
40	-0.087	-2.289	0.434	0.112	0.632	0.540	2.318	-1.169	-1.383	1.178	-0.415	-0.385	-1.617	1.755	-0.920
45	1.268	-1.955	-0.244	0.790	-0.122	-0.173	2.701	-1.141	0.480	0.957	-0.380	-0.246	-2.965	1.470	-1.180
50	-0.095	-2.556	0.407	0.180	0.723	0.576	2.699	-1.286	-1.415	1.231	-0.410	-0.405	-1.846	1.989	-1.079
55	1.133	-2.109	-0.251	0.811	-0.011	-0.113	2.919	-1.192	0.409	0.933	-0.371	-0.211	-2.958	1.579	-1.274
60	-0.072	-1.462	0.532	-0.097	0.353	0.406	1.118	-0.772	-1.256	1.022	-0.417	-0.316	-0.878	1.033	-0.411
65	1.275	-1.355	-0.193	0.634	-0.316	-0.287	1.826	-0.855	0.564	0.828	-0.388	-0.196	-2.437	0.961	-0.822
70	1.259	-2.487	-0.303	0.931	0.050	-0.097	3.464	-1.387	0.405	1.053	-0.375	-0.286	-3.420	1.924	-1.509
75	1.280	-1.339	-0.187	0.632	-0.319	-0.285	1.811	-0.851	0.561	0.827	-0.393	-0.196	-2.422	0.942	-0.808
80	1.265	-2.163	-0.262	0.843	-0.054	-0.142	3.004	-1.236	0.450	0.999	-0.376	-0.263	-3.142	1.645	-1.303
85	-0.069	-1.500	0.528	-0.085	0.368	0.413	1.173	-0.790	-1.260	1.027	-0.418	-0.317	-0.913	1.068	-0.432
90	1.264	-2.295	-0.286	0.882	-0.012	-0.126	3.194	-1.303	0.426	1.020	-0.379	-0.274	-3.260	1.758	-1.395

Table 9: Per-Atom Energies of Pent-2-ene at Different Timesteps (Hartree)

From Table 9, the pent-2-ene dataset introduces an additional carbon atom (C5), which fundamentally alters the electronic distribution and introduces new long range interactions within the 10 Å cutoff. Since deepset fails to adjust for this extended conjugation, significantly underestimating the energy of each conformer (illustrated in Figure 17), it is understood that the model is still reliant on interpolative learning. The atom energies change due to stretching and bending through time (conformers), as seen with the per-atom energies of carbon which vary from -2.556 to 1.280 hartree. However, across the steps, the majority of energy changes are subtle and perform as expected, inline with the theoretical chemistry behind alkene conformers of small torsional and vibration distortions causing changes through conformers.

Two notable data values are Step 5 and 15, both of which were selected due to having highly similar predictions for per-atom energies. Using the total energies for these values, it is possible to evaluate the accuracy of these predictions and the model's ability to predict per-atom energies. Table 10 details the comparison between the reference energies (but-2-ene dataset) and the predicted per atom energies derived from the model. Step 40, which substantially differs in the per atom energies, is included as a control to assess the specificity of the model's agreement with Steps 5 and 15. If the values align closely with the reference values for Steps 5 and 15 but not for Step 40, then the prediction can be drawn that the model reliably captures physically meaningful energy patterns rather than coincidental matches.

Step	**Outputs (Predicted)**	**Labels (True)**
5	-12.6984	-12.6952
15	-12.7168	-12.7201
40	-12.7123	-12.7157

Table 10: Predicted and true total energies at selected steps for but-2-ene (Hartree).

Based on Table 10, it is evident that there is no statistically significant correlation between the per atom energies of steps 5 and 15. Crucially, the true value for step 40 lies between those of 5 and 15. In a flawlessly generalisable model the predicted per atom energies would also be expected to fall between this range, however, this was not observed.

A possible explanation for this lies within the mixture of experts (MoE) gating mechanism used in deepset. In such MoE models, small changes in geometry can cause discontinuous shifts in expert weights, leading to sudden changes (jumps) in the predicted potential energy surface (PES). These jumps appear either as crossovers, where the dominant expert suddenly switches, or as transitions to entirely different experts, although more common with asymmetric gating. As a result of this, in geometrically similar conformers deepset predicts inconsistent potential energies, which contradicts the expected smoothness of a typical PES. These discontinuities are especially problematic for force predictions, since forces are calculated by the negative derivative of the energies. Consequently, small inconsistencies in the energy predictions increase in force loss. This is consistent with the training metrics, where the force remained stubbornly high compared to the loss for energies. Additionally, cutoff induced discontinuities may add to the instability when combined with expert switching, although unlikely in this case due to the small molecules and large cutoff.

To gain a better understanding of these potential issues and their impact in deepset, it would be informative to test the model using a single expert (removing the MoE mechanism). This alteration would be expected to give a smoother PES, and identify whether the weighting mechanism is the root of the issue.

5 Conclusion

Overall, the objective of this report was to improve the accuracy of energy and force predictions by the Deepset model during training and to test its generalisability to unseen data. It was observed that the addition of scalar features, such as cubic and square root transformations, provided clear benefits, while the integration of GNN layers and expansion of the RBF representation increased loss, demonstrating that greater architectural complexity does not always enhance modelling performance and can lead to oversmoothing of learned representations. A recurring issue in molecular dynamics is the difficulty of generalising representations beyond the training dataset, as evidenced by the significant drop in performance when evaluating pent-2-ene conformers. Analysis of per-atom energies further reinforced this understanding, emphasising the importance of chemically meaningful node representations in permutation-invariant models.

Future research should focus on expanding chemical diversity within training datasets, exposing models to a wider range of bonding patterns and torsional angles through datasets such as ANI-1 and MD17,[40;41] in order to better equip models for generalisation to unseen molecular systems and improve their extrapolative capabilities. As computational chemistry moves closer to developing broadly generalisable models, studies such as this help clarify which architectural choices enable transferable representations and where current approaches fall short in capturing chemically meaningful variation.

References

[1] A. Dongare and R. Kharde and A. Kachare, *Introduction to Artificial Neural Network*, 2012, vol. 2, pp. 189–192.

[2] Weights and Biases, *ReLU vs Sigmoid Function in Deep Neural Networks*, `https://wandb.ai/ayush-thakur/dl-question-bank/reports/ReLU-vs-Sigmoid-Function-in-Deep-Neural-Networks--VmlldzoyMDk0MzI`, (accessed March 2025).

[3] S. Hayman, *The McCulloch-Pitts model*, 1999, vol. 6, pp. 4438–4439.

[4] J. Gasteiger and J. Zupan, *Angewandte Chemie International Edition in English*, 1993, **32**, 503–527.

[5] Z. Yang, X. Zeng and Y. Zhao, *Signal Transduction and Targeted Therapy*, 2023, **8**, 115.

[6] S. R. Dubey, S. K. Singh and B. B. Chaudhuri, *A Comprehensive Survey and Performance Analysis of Activation Functions in Deep Learning*, 2021, `https://arxiv.org/abs/2109.14545`.

[7] PyTorch, *SiLU*, 2025, `https://docs.pytorch.org/docs/stable/generated/torch.nn.SiLU.html`, (accessed May 2025).

[8] S. Elfwing, E. Uchibe and K. Doya, *Sigmoid-Weighted Linear Units for Neural Network Function Approximation in Reinforcement Learning*, 2017, `https://arxiv.org/abs/1702.03118`.

[9] GeeksforGeeks, *ML — Stochastic Gradient Descent (SGD)*, 2025, `https://www.geeksforgeeks.org/ml-stochastic-gradient-descent-sgd`, (accessed March 2025).

[10] PyTorch, *Adam*, 2025, `https://pytorch.org/docs/stable/generated/torch.optim.Adam.html`, (accessed April 2025).

[11] A. Vaswani, N. Shazeer and N. Parmar, *Attention Is All You Need*, 2023, `https://arxiv.org/abs/1706.03762`.

[12] J. Jiang, L. Ke and L. Chen, *Transformer technology in molecular science*, vol. 14, p. e1725.

[13] A. Sultan, J. Sieg and M. Mathea, *Transformers for molecular property prediction: Lessons learned from the past five years*, 2024, `https://arxiv.org/abs/2404.03969`.

[14] S. Liu, Y. Wang and T. Wang, *Improved Drug-target Interaction Prediction with Intermolecular Graph Transformer*, 2021, `https://arxiv.org/abs/2110.07347`.

[15] P. Thölke and G. D. Fabritiis, *TorchMD-NET: Equivariant Transformers for Neural Network based Molecular Potentials*, 2022, `https://arxiv.org/abs/2202.02541`.

[16] S. Mostafa and F.-X. Wu, *Diagnosis of autism spectrum disorder with convolutional autoencoder and structural MRI images*, Academic Press, 2021, pp. 23–38.

[17] T. Hung, Z. Xu and D. Kang, *The Journal of Physical Chemistry C*, 2022, **126**, 2813–2822.

[18] B. L. Douglas, *The Weisfeiler-Lehman Method and Graph Isomorphism Testing*, 2011, `https://arxiv.org/abs/1101.5211`.

[19] R. Mehrotra and K. Guo, *Predicting Molecular Properties with Graph Attention Networks*, 2020, `https://cs230.stanford.edu/projects_winter_2020/reports/32642951.pdf`.

[20] TorchMD-Net, *Welcome to the TorchMD-NET Documentation*, 2025, `https://torchmd-net.readthedocs.io/en/latest/`, (accessed March 2025).

[21] J. Wang, R. M. Wolf and J. W. Cadwell, *Journal of Computational Chemistry*, 2004, **25**, 1157–1174.

[22] TorchMD, *TorchMD About GitHub*, 2025, `https://github.com/torchmd/torchmd/blob/main/README.md`, (accessed March 2025).

[23] S. Doerr, M. Majewski and Pérez, *TorchMD: A Deep Learning Framework for Molecular Simulations*, American Chemical Society (ACS), 2021, vol. 17, p. 2355–2363.

[24] Y. Cao, B. Chen and X. Li, *Force Matching with Relativistic Constraints: A Physics-Inspired Approach to Stable and Efficient Generative Modeling*, 2025, `https://arxiv.org/abs/2502.08150`.

[25] P. Xu, X. Mou and Q. Guo, *Coarse-Grained Molecular Dynamics Study based on TorchMD*, vol. 34, pp. 957–969.

[26] R. P. Pelaez, G. Simeon and Galvelis, *TorchMD-Net 2.0: Fast Neural Network Potentials for Molecular Simulations*, 2024, `http://dx.doi.org/10.1021/acs.jctc.4c00253`.

[27] M. Artexe, S. Bhosale and N. Goyal, *Efficient Large Scale Language Modeling with Mixtures of Experts*, 2022.

[28] N. Karimi, S. Kazem and D. Ahmadian, *On a generalized Gaussian radial basis function: Analysis and applications*, 2020, vol. 112, pp. 46–57.

[29] A. van Ooyen, *New Approaches for the Generation and Analysis of Microbial Typing Data*, Elsevier Science, 2001, p. 38.

[30] C. Xu, Y. Lu and X. Deng, *Prediction of Molecular Conformation Using Deep Generative Neural Networks*, 2023, vol. 41, pp. 3684–3688.

[31] K. V. Chuang and M. J. Keiser, *Attention-Based Learning on Molecular Ensembles*, 2020, `https://arxiv.org/abs/2011.12820`.

[32] A. M. Schweidtmann, J. G. Rittig and J. M. Weber, *Physical pooling functions in graph neural networks for molecular property prediction*, Elsevier BV, 2023, vol. 172, p. 108202.

[33] M. W. X. Liu, Y. Zhang, *Revisiting Edge Perturbation for Graph Neural Network in Graph Data Augmentation and Attack*, 2024, `https://arxiv.org/abs/2403.07943`.

[34] P. Veličković, G. Cucurull, A. Casanova and A. Romero, *Graph Attention Networks*, 2018, `https://arxiv.org/abs/1710.10903`.

[35] B. Scholkopf, S. Mika, C. Burges and P. Knirsch, *IEEE Transactions on Neural Networks*, 1999, **10**, 1000–1017.

[36] X. Ying, *An Overview of Overfitting and its Solutions*, `https://doi.org/10.1088/1742-6596/1168/2/022022`, 2019, Journal of Physics: Conference Series.

[37] C. Bishop, *Pattern Recognition and Machine Learning*, Springer, 2006, pp. 257–267.

[38] M. Zaheer, S. Kottur and S. Ravanbakhsh, *Deep Sets*, 2018, `https://arxiv.org/abs/1703.06114`.

[39] W. Ong, P. Kirubakaran and J. Karanicolas, *Poor Generalization by Current Deep Learning Models for Predicting Binding Affinities of Kinase Inhibitors*, 2023, `https://doi.org/10.1101/2023.09.04.556234`.

[40] J. Smith, O. Isaye and A. E. Roitberg, *Chemical Science*, 2017, **8**, 3192–3203.

[41] A. S. Christensen and O. A. von Lilienfeld, *On the role of gradients for machine learning of molecular energies and forces*, 2020, `https://arxiv.org/abs/2007.09593`.