Bachelorarbeit

Neural Graph Fingerprints vs. Extended Connectivity Fingerprints - Structural Similarity and Model Performance

Verfasser

Albert Dinstl

angestrebter akademischer Grad

Bachelor of Science (BSc)

Wien, 2025

Studienkennzahl lt. Studienblatt:	A 033 521
Fachrichtung:	Informatik
Betreuer:	Assoz. Prof. Dipl.-Inf. Dr. Nils Morten Kriege

Contents

Abstract

This thesis investigates the structural similarity between Neural Graph Fingerprints (NGF) and Extended Connectivity Fingerprints (ECFP). This was achieved by reimplementing the NGF model in the PyTorch Geometric framework to conduct a critical analysis of the "large weights" hypothesis for ECFP approximation. The results show that using large weights fails to improve structural alignment with ECFP in terms of pairwise distance correlation. This work also demonstrates two other key findings: NGF embeddings more closely resemble Count ECFP than Binary ECFP, and the ECFP algorithm described in the original NGF paper differs from standard implementations like RDKit. The thesis concludes by exploring architectural adaptations, finding that periodic activation functions (e.g., sine) and edge-conditioned aggregation can enhance either structural similarity or model performance.

1 Introduction

Molecular property prediction plays a crucial role in areas like material design and drug discovery. A central challenge to this discipline is finding representations of molecular graphs that are well suited to be an input to machine learning models. Traditionally, this area was dominated by molecular fingerprints like Extended Connectivity Fingerprints (ECFP) due to their computational efficiency [4]. Recently, advancements in deep learning have shifted the attention towards learned, Graph Neural Network-based approaches. Among those, Neural Graph Fingerprints (NGF), first introduced by Duvenaud et. al [6] in 2015 try to unite these two approaches by attempting to approximate ECFP's structure in an end-to-end differentiable, learnable manner. Although this approach was a very significant contribution to the field, some of its theoretical foundations and practical details remain insufficiently specified. This thesis critically evaluates the NGF approach through a systematic, investigation. The work begins with a from-scratch reimplementation of the core algorithms to faithfully test the claims made in the original paper. This process uncovers inconsistencies in the baseline comparisons and allows for a controlled analysis of the model's components. The following empirical experiments are designed to dissect the "large weights" hypothesis and to explore a range of architectural modifications, from alternative activation functions to different neighborhood aggregation schemes. Through this analysis, the thesis clarifies the relationship between NGF and ECFP and identifies specific model adaptations that yield significant improvements in both structural alignment and downstream task performance.

2 Scientific Background

In molecular property prediction, it is crucial to develop effective molecular representations, commonly known as molecular fingerprints, to serve as inputs for machine learning models. The traditional paradigm in this area is the discrete and rule based ECFP algorithm [4]. Another approach is using Graph Neural Networks (GNN) which apply a data driven deep learning strategy to learning powerful molecular representations [8, 26, 10]. A third strategy is Neural Graph Fingerprints (NGF), which blend ECFP with the adaptive learning capabilities of GNNs [6].

2.1 ECFP

Extended Connectivity Fingerprints are widely used for encoding molecular substructures into fixed-length vectors. However there are several variations of ECFP with different characteristics which could affect downstream task performance. While all ECFP variants share the core principle of aggregating neighbourhood information and iteratively hashing local atomic neighbourhoods. They differ in the way they aggregate and represent these hashed values [4].

2.1.1 The ECFP Algorithm

The ECFP Algorithm consists of three central steps:

Assigning initial atom identifiers: Tuples are created based on atom features which are hashed to integer identifiers [4].

Iteratively updating hashed identifiers: For each atom, the identifier is updated using information from neighbouring atoms, as well as bond order information of these neighbours and hashed again. This is repeated until a predefined number of iterations (the radius) has been performed [4].

Folding identifiers into a fingerprint array: All integer identifiers from all iterations are folded into a bit array using the modulo operation to determine the final position of a substructure identifier in the fingerprint array [4].

2.1.2 Binary ECFP

Binary ECFP is the most commonly used variant of ECFP [25]. Each bit in the fixed-length vector indicates either the presence (1) or the absence (0) of a molecular substructure. The result is a sparse binary vector, generally of length 1024 or 2048 which encodes the presence or absence of substructures without their frequency [4].

2.1.3 Count ECFP

Unlike binary ECFP, count ECFP tracks the frequency of molecular substructures. Instead of just marking a substructure's presence, it increments a counter for each occurrence, resulting in an integer vector where elements can be greater than one [4, 17].

2.2 NGF

In their paper "Convolutional Networks on Graphs for Learning Molecular Fingerprints" Duvenaud et. al proposed a novel framework for learning molecular representations from graph structured data. This approach known as Neural Graph Fingerprints aims at replicating Extended Connectivity Fingerprints [4] in an end-to-end trainable manner.

ECFP relies on discrete operations such as hashing to encode structural information about molecules which makes them unable to be tailored to specific tasks and datasets. NGF tries to address this limitation by replacing these non-differentiable operations with trainable components thereby allowing for learning through gradient-based optimization.

NGF operates on molecular graphs in which nodes represent atoms which are described by a feature vector containing information about its type, degree, valence etc. and edges represent bonds which may also include bond information.

Duvenaud et. al observed that even without training, NGF can produce meaningful molecular representations. They claim that even a randomly initialized NGF with large weights can produce similar embeddings as ECFP in terms of pairwise molecular distance. For this, they used a continuous generalization of the Tanimoto distance measure: [6]

$$distance(x, y) = 1 - \sum \min(x_i, y_i) / \sum \max(x_i, y_i) \tag{1}$$

The components of the NGF Algorithm are specifically designed to mimic the inner workings of ECFP. [6, 4]

Smooth function: A smooth non-linearity like the hyperbolic tangent tanh is applied to the node aggregation which should mimic the thresholding effect of hash functions. The authors claim that in the limit of large weights, tanh approximates a step function which can act as a soft hash.

Sparsify function: To mimic the non-differentiable operation of bit setting. NGF uses softmax as a function to sparsify the embeddings.

Permutation invariant neighborhood aggregation: In ECFP the order in which atoms and bonds are ordered does not change the output. NGF shares that property by employing operations such as summation when aggregating neighborhood information.

2.2.1 Large weights

The authors of the original NGF paper repeatedly mention the use of large weights in their NGF. This section analyzes the effect of using weights of large magnitude from a theoretical standpoint. [6]

Let $W_{neigh}, W_{self}, W_{fp}$ be the randomly initialized weight matrices of our NGF and f_d the dimension of the final fingerprint. We aggregate a node embedding r_{aggr} by applying W_{self} to the atom r and W_{neigh} to the sum of its neighbours $r_1...r_n$, $n \leq 4$ (in organic molecules).

$$r_{aggr} = W_{self} r_a + \sum_{i=1}^{n} W_{neigh} r_n$$

This then gets passed to the hyperbolic tangent $tanh(r_{aggr})$, which if the weights are large enough, outputs a vector of dimension containing only entries equal to -1 or 1.

$$\tanh(r_{aggr})_i = \begin{cases} 1 & \text{if } r_{aggr} > 0 \\ -1 & \text{if } r_{aggr} < 0 \end{cases}$$

This is then projected into fingerprint space using W_{fp}. Note that if the entries of W_{fp} are large enough, each column of W_{neigh} is essentially drawn from a zero-mean Gaussian with large variance σ^2. This produces a vector of dimension f_d and large positive and negative values of high variance.

$$r_{projected} = W_{fp} \tanh(r_{aggr})$$

Due to the values of this vector being large random values of high variance, softmax essentially behaves like argmax.

$$r_{sparsified} = \text{softmax}(r_{projected})$$

Hence, $r_{sparsified}$ is likely to only have one bit set, thereby aligning NGF embeddings and Extended Connectivity Fingerprints in sparsity [6].

3 Related Work

Extended Connectivity Fingerprints (ECFP), first introduced by David Rogers and Matthew Hahn in 2010 [4], remain widely used in molecular property prediction. Their effectiveness as input features for machine learning models has been demonstrated in several studies [17, 2], and their computational efficiency has made them a popular benchmark for evaluating more complex representation learning methods [25, 26].

In 2015, Duvenaud et al. [6] proposed a neural network-based molecular fingerprinting approach designed to mimic the discrete, rule-based structure of ECFP in a differentiable way. Building on this idea, Kearnes et al. [8] proposed a more flexible architecture that moves beyond mimicking ECFP, aiming instead to learn task-specific molecular representations without adhering to the fingerprinting paradigm.

Yang et al. [26] provide a broad comparative evaluation of learned molecular representations-including message passing networks-against traditional approaches such as ECFP. More recently, Graph Neural Networks have become a dominant framework in molecular representation learning, demonstrating strong performance across a variety of property prediction tasks [19, 3, 22].

4 Research Questions

This thesis is centered around five main questions:

4.1 Effect of large weights

Duvenaud et. al often mention the use of large random weights when comparing NGF embeddings to ECFP. However, the magnitude of these weights is never mentioned and aside from theoretically aligning NGF with ECFP in sparsity, it is unclear how large weights affect pairwise distance correlation to ECFP and downstream task performance. This thesis aims to fill in those gaps by empirically dissecting the effects of the individual components of the NGF algorithm on intermediate node and graph representations [6].

4.2 Similarity of Algorithm 1 to ECFP

Duvenaud et. al provide an algorithmic description of ECFP which is compared to the NGF algorithm. This description omits crucial steps compared to official implementations of ECFP. This thesis aims to analyze the effect of these differences [6].

4.3 Similarity of NGF to ECFP

Duvenaud et. al compare NGF embeddings to Binary ECFP. In this thesis NGF embeddings will also be compared to Count ECFP [6, 4].

4.4 Methodological Enhancements

This thesis explores methodological enhancements of the NGF algorithm [6] by exchanging components of the NGF algorithm for potentially more capable alternatives. This will be done by conducting experiments on downstream tasks and closely monitoring structural similarity to ECFP.

4.5 Existing Implementation in PyTorch Geometric

NGF has been implemented across muliple modern machine learning frameworks [9, 16]. This thesis explores the use of the existing implementation in PyTorch Geometric to recreate the findings by Duvenaud et. al [6].

5 Methods

5.1 Datasets

The datasets were processed in memory to allow for efficient experimentation. Each molecule was initially represented as a SMILES string [24] which served as a starting point for constructing representations required by the different fingerprinting methods.

5.1.1 BACE

The BACE dataset contains binary classification data for 1513 compounds potentially binding to human β-secretase 1 (BACE 1) which is an important target in Alzheimers disease research. The binary classification task is to predict whether a molecule is an active inhibitor or not [14].

5.1.2 ESOL

The ESOL dataset contains water solubility measurements for 1128 small organic molecules. This task is a regression problem aiming at predicting the log solubility in mols per litre of a compound [5].

5.1.3 Lipophilicity

The lipophilicity dataset contains experimental values for the logD at pH 7.4 of 4200 molecules. Similar to ESOL this is a regression task aimed at predicting this value [1, 23].

5.2 Pairwise Distance Correlation

To recreate the pairwise distance correlation plot in the original NGF paper [6] and to conduct further experiments examining the structural similarity of NGF embeddings to the variants of ECFP. The distance metric described in equation 1 will be used.

5.3 Downstream Tasks

The downstream tasks corresponding to the datasets above will be implemented using a Multi-Layer-Perceptron (MLP) for regression or classification, depending on the dataset at hand. Depending on the experiment, inputs will be ECFP fingerprints or NGF embeddings. When an NGF is trained end-to-end the MLP will serve as a projection head that makes predictions on learned representations.

5.4 Experimental Setup

The experimental evaluation of this work was conducted in three distinct stages, designed to first establish baselines, then critically test the core claims of the original NGF paper, and finally explore potential architectural improvements.

5.4.1 Stage 1: Baseline Establishment

The first stage focused on establishing performance benchmarks for both traditional and neural fingerprint methods. The following five molecular representations were evaluated on the three downstream tasks:

- Binary ECFP (RDKit implementation [11])
- Count ECFP (RDKit implementation [11])
- Algorithm 1 from NGF paper [6](Reimplementation from scratch)
- The baseline NGF from this work (frozen and end-to-end trained)
- The existing NGF implementation in PyTorch Geometric (frozen and end-to-end trained)

For all fingerprint types, the fingerprint dimension was fixed at 2048 and the radius at 2, corresponding to two layers in the NGF architecture. The primary metrics were downstream task performance (ROC-AUC or RMSE) and, for ECFP variants, their inherent bit sparsity.

5.4.2 Stage 2: Investigating the 'Large Weights' Hypothesis

The second stage was designed to empirically test the claim that large random weights align NGF embeddings with ECFP. NGF models were initialized with random weights scaled by several orders of magnitude (from 0.1 to 1000). To dissect the effects of this scaling, the following metrics were collected for the resulting frozen embeddings:

- **Bit Sparsity:** To directly quantify the effect of weight scaling on the embedding structure.
- **Pairwise Distance Correlation:** To measure the structural alignment with Binary ECFP.
- **Frozen Downstream Task Performance:** To assess the practical utility of the embeddings without training.

5.4.3 Stage 3: Exploring Architectural Improvements

The third stage explored the impact of individual architectural components. The default summation, activation (tanh), and sparsification (softmax) functions of the NGF were systematically replaced with alternatives. For each variation, a comprehensive set of metrics was collected to evaluate the trade-offs between learnability and structural fidelity:

- Frozen and end-to-end downstream task performance
- Frozen and end-to-end correlation in pairwise distances to ECFP

- Bit-Sparsity: This metric was collected for variants using alternative sparsification functions to directly compare their sparsity-inducing properties against the baseline and the effects of weight scaling from Stage 2.

For all downstream task evaluations across all stages, an MLP classifier or regressor was used, depending on the task.

5.5 Featurization

In the featurization step, it was crucial to ensure that all fingerprinting methods were using the same subset of atom and bond features as in the original NGF paper. The following one hot encoded atom features were used: [6]

- Atom symbol
- Degree
- Number of attatched hydrogens
- Implicit valence
- Aromaticity

The bond features were a concatenation of:

- Bond type $\in \{single, double, triple\}$
- Conjugation
- Ring membership

6 Implementation

The implementation of the NGF algorithm and the training pipeline was carried out in PyTorch Geometric [15]. For ECFP computation the cheminformatics library RDKit [11] was used.

6.1 Algorithm 1

As a next step the algorithmic description of ECFP depicted in the paper by Duvenaud et. al [6] was reimplemented. RDKit [11] was used to gather atom features. Compared to official implementations of ECFP and the original ECFP paper [4] this version does not hash bond order information and iteration depth.

6.2 Training Pipeline

The figure below 1 illustrates the experimental pipeline that was used. Starting from SMILES strings [24] either RDKit molecule or PyTorch Geometric Data objects are being constructed. These get passed to an NGF and an ECFP variant which result in a fingerprint-matrix or embedding-matrix respectively. Afterwards, these embeddings and fingerprints are used as input to an analysis of their correlation of their pairwise distances using the continuous generalization of the Tanimoto distance metric introduced in equation 1 and to an MLP which evaluates the embeddings on a downstream task. [11, 15]

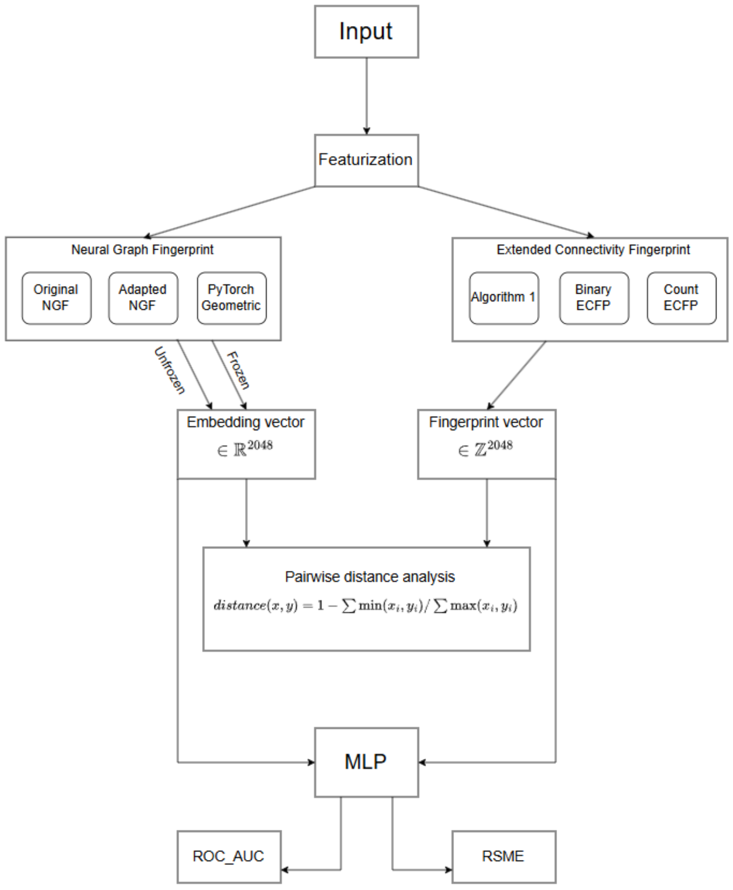

Input
Featurization
Neural Graph Fingerprint
Original NGF
Adapted NGF
PyTorch Geometric
Extended Connectivity Fingerprint
Algorithm 1
Binary ECFP
Count ECFP
Unfrozen
Frozen
Embedding vector $\in \mathbb{R}^{2048}$
Fingerprint vector $\in \mathbb{Z}^{2048}$
Pairwise distance analysis
$distance(x, y) = 1 - \sum \min(x_i, y_i) / \sum \max(x_i, y_i)$
MLP
ROC_AUC
RSME

Figure 1: **Overview of the training and evaluation pipeline.** Molecular representations are generated from SMILES strings using either traditional ECFP algorithms or a Neural Graph Fingerprint (NGF) model. The resulting fingerprint or embedding vectors are used for two purposes: 1) training an MLP for downstream prediction tasks (e.g., classification or regression), and 2) a pairwise distance analysis to correlate their structural similarity.

6.3 Neural Graph Fingerprints

The main goal of this implementation was to faithfully reproduce the algorithm described in the original paper while also leaving room for exchanging different components of the algorithm. This implementation allowed for a controlled exploration of the contributions of the individual components while also being able to exchange them for possibly more capable alternatives.

7 Results

7.1 Baseline Experiments

Establishing consistent baselines is important for understanding the relative performance of different fingerprinting methods. Evaluating both the different ECFP implementations as well as the original NGF architecture provides a reference point from which to assess the architectural modifications in further experiments.

7.1.1 Binary ECFP

In this first baseline experiment, the goal was to get an idea of how well Binary ECFP, the most common variation of ECFP, performs in all three of the downstream tasks.

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8646
Regression	ESOL	RMSE = 1.1919
Regression	Lipophilicity	RMSE = 0.8827

Table 1: Performance of Binary ECFP across datasets

Dataset	**Metric (Value)**
BACE	Bit Sparsity = 0.9707
ESOL	Bit Sparsity = 0.9896
Lipophilicity	Bit Sparsity = 0.9766

Table 2: Bit Sparsity of Binary ECFP

7.1.2 Count ECFP

In the second baseline experiment, Binary ECFP was exchanged for Count ECFP. Note that there are substantial improvements in both the regression tasks and a marginal improvement in the classification task. This suggests that count fingerprints are more suitable for learning.

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8668
Regression	ESOL	RMSE = 0.8704
Regression	Lipophilicity	RMSE = 0.7700

Table 3: Performance of Count ECFP across datasets

Dataset	**Metric (Value)**
BACE	Bit Sparsity = 0.9707
ESOL	Bit Sparsity = 0.9896
Lipophilicity	Bit Sparsity = 0.9766

Table 4: Bit Sparsity of Count ECFP

7.1.3 Algorithm 1

In this experiment, the goal was to examine the downstream task performance and bit sparsity of ECFP fingerprints generated by Algorithm 1 of the NGF paper.

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8484
Regression	ESOL	RMSE = 1.1646
Regression	Lipophilicity	RMSE = 0.8875

Table 5: Performance of Algorithm 1 (radius=2) across datasets

Dataset	**Metric (Value)**
BACE	Bit Sparsity = 0.9717
ESOL	Bit Sparsity = 0.9895
Lipophilicity	Bit Sparsity = 0.9772

Table 6: Bit Sparsity of Algorithm 1

7.1.4 Baseline NGF

The baseline NGF model, implemented for this thesis using tanh and softmax, successfully reproduces the key characteristics described in the original paper [6].

In its frozen, randomly initialized state, the model already exhibits a high degree of structural similarity to Binary ECFP. This is illustrated in the top row of figure 2. The frozen model achieves a respectable ROC-AUC of 0.8408 on BACE and outperforms the standard Binary ECFP on both regression tasks (table 7).

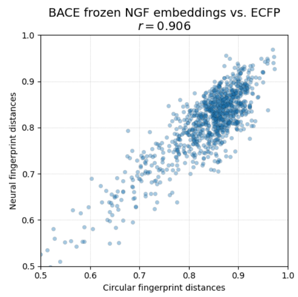
When trained, a significant improvement in performance is observed across all three datasets, as shown in table 8.

Task	Dataset	Metric (Value)
Binary Classification	BACE	ROC_AUC = 0.8408
Regression	ESOL	RMSE = 0.9530
Regression	Lipophilicity	RMSE = 0.8128

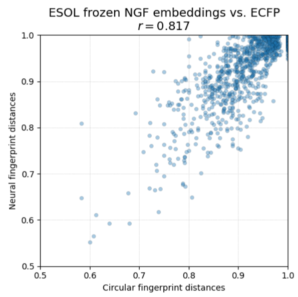
Table 7: NGF with tanh and softmax frozen downstream task performance

Task	Dataset	Metric (Value)
Binary Classification	BACE	ROC_AUC = 0.8627
Regression	ESOL	RMSE = 0.8009
Regression	Lipophilicity	RMSE = 0.6671

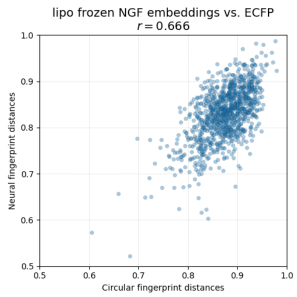
Table 8: NGF with tanh and softmax end-to-end trained downstream task performance



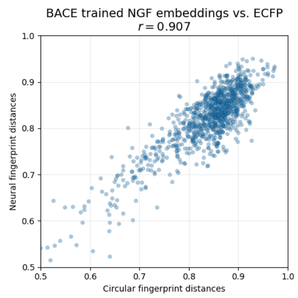
(a) BACE



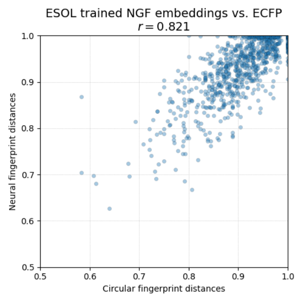
(b) ESOL



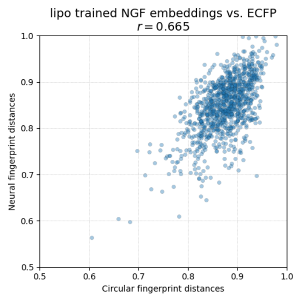
(c) Lipophilicity



(d) BACE



(e) ESOL



(f) Lipophilicity

Figure 2: Structural correlation of the baseline NGF embeddings to Binary ECFP. The top row shows the performance of the frozen model, while the bottom row shows the same model after end-to-end training.

7.1.5 Existing Implementation in PyTorch Geometric

The results for the frozen, randomly initialized PyTorch Geometric model [16] show a significant deviation from the behavior described in the original NGF paper. As detailed in table 9, its frozen downstream task performance was markedly poor, with a ROC-AUC of just 0.6593 on BACE and very high RMSE values on the regression tasks. This underperformance was mirrored in its structural alignment (figure 3). The pairwise distance correlation to ECFP, shown in Figure 3, was extremely low across all datasets, indicating that the frozen embeddings capture very little of ECFP's structural information.

However, the model's performance improved substantially after end-to-end training. Table 10 shows that the trained model achieved results comparable the reimplemented trained variant.

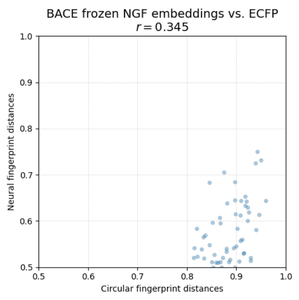
This contrast between the ineffective frozen model and the competitive trained model suggests that the PyTorch Geometric implementation does not replicate the key finding of the original paper, where frozen embeddings were shown to be powerful predictors. This implies a potential difference in the underlying implementation details compared to the one proposed in this thesis and the original paper [6].

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.6593
Regression	ESOL	RMSE = 1.7478
Regression	Lipophilicity	RMSE = 1.2369

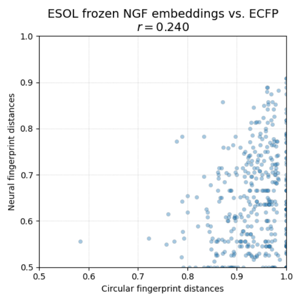
Table 9: Frozen performance of PyTorch Geometric NGF across datasets

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8434
Regression	ESOL	RMSE = 0.8122
Regression	Lipophilicity	RMSE = 0.6543

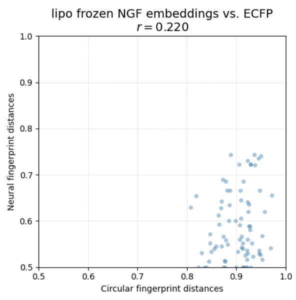
Table 10: End-to-end performance of PyTorch Geometric NGF across datasets



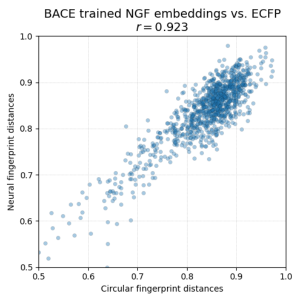
(a) BACE



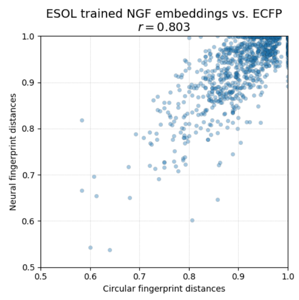
(b) ESOL



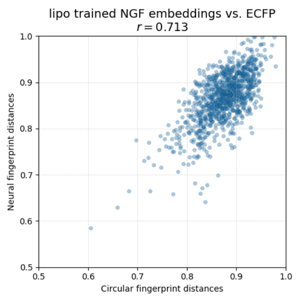
(c) Lipophilicity



(d) BACE



(e) ESOL

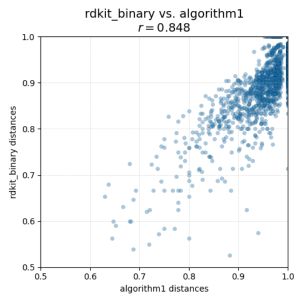


(f) Lipophilicity

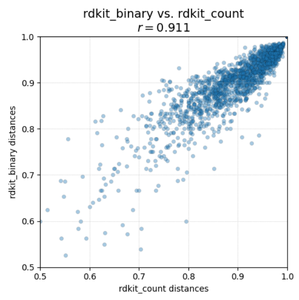
Figure 3: Correlation of pairwise distances between frozen and trained NGF embeddings vs Binary ECFP

7.2 Algorithm 1 vs. RDKit

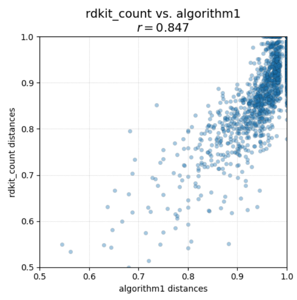
To investigate how these differences affect the similarity between these variations of ECFP, the correlation of their pairwise distances using the generalized continuous Tanimoto distance metric described in equation 1 was used. The fingerprint dimension was chosen at 2048 and the radius at 2. The results can be seen in the figure 4 below.



(a) ESOL



(b) ESOL



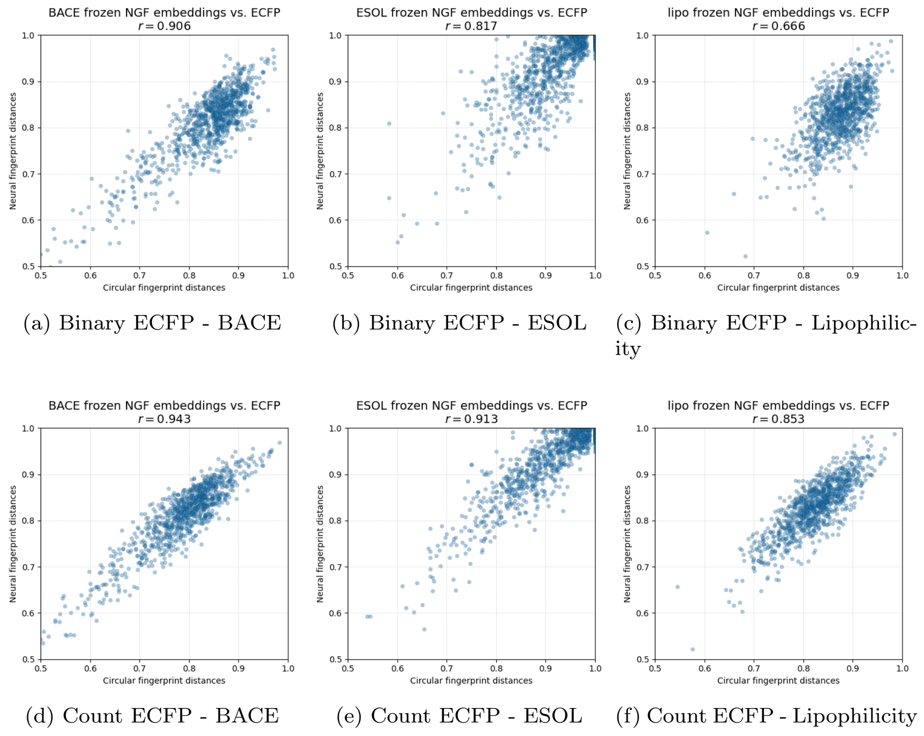
(c) ESOL

Figure 4: Correlation of pairwise distances between Binary ECFP, count ECFP and Algorithm 1

The three plots (figure 4) above show the comparison of the pairwise distance between the ECFP variants. The comparison between RDKits count fingerprint and binary fingerprint yielded the highest correlation ($r = 0.911$), indicating that despite their difference in encoding frequency versus presence of a substructure, they capture very similar structural relationships between molecules. In contrast, the correlation between algorithm 1 from the NGF paper and RDKits fingerprinting methods is lower, at $r = 0.848$ for RDKits binary fingerprint and $r = 0.847$ for RDKits count fingerprint. This suggests that although algorithm 1 approximates ECFP reasonably well, the deviations from the original ECFP algorithm lead to measurable differences in the fingerprint space.

7.3 NGF's similarity to ECFP

Neural Graph Fingerprints with the hyperbolic tangent smooth function and softmax as described by Duvenaud et. al [6] exhibit more similarity to Count ECFP than to Binary ECFP.



(a) Binary ECFP - BACE (b) Binary ECFP - ESOL (c) Binary ECFP - Lipophilicity

(d) Count ECFP - BACE (e) Count ECFP - ESOL (f) Count ECFP - Lipophilicity

Figure 5: Structural similarity of the frozen baseline NGF to Binary and Count ECFP.

7.4 Large random weights

7.4.1 Toy Example: Ethanol

As a preliminary experiment on the effect of scaling weights by different magnitudes, the embedding of a single molecule was investigated. For this, the molecule ethanol C_2H_6O was used. The weights of a randomly initialized NGF were scaled by different magnitudes and the internal components of the NGF were monitored during the computation of a 2048-dimensional fingerprint for the ethanol molecule. The follwing metrics were monitored:

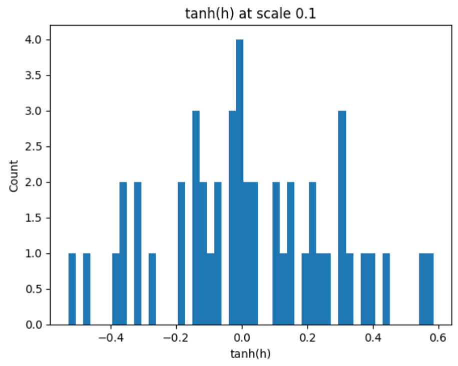
- Saturation of the hyperbolic tangent $\tanh(x)$ showing the relative frequency of the activation having an absolute value of 1
- Peak of the distribution resulting of the softmax function
- Shannon entropy of distribution resulting from the softmax function [18]

For ethanol, this yielded the following results:

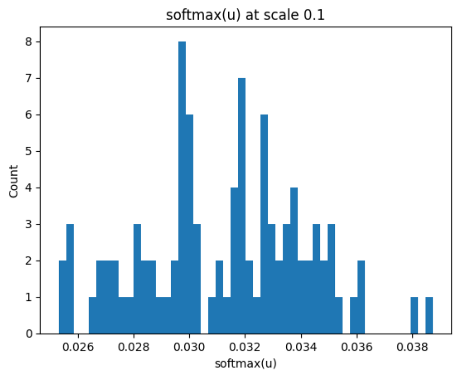
Weight scale	**tanh-saturation**	**Softmax Peak**	**Softmax Entropy**
0.1	0.000	0.001	7.573
1	0.404	0.857	0.562
10	0.935	0.980	0.076
100	0.995	1.000	0.000
1000	1.000	1.000	0.000

Table 11: Observed metrics for randomly initialized NGF across different weight scales

To get an idea of how the activations of tanh and the results of the softmax function are distributed, this experiment was repeated with a fingerprint dimension of 32 so that meaningful plots could be created.

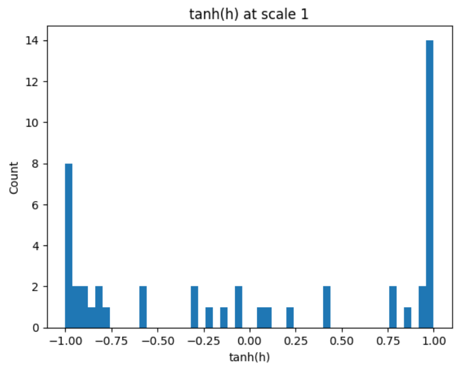


(a) Hyperbolic tangent

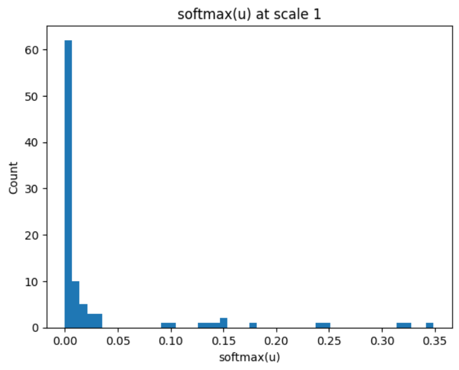


(b) Softmax

Figure 6: Result distributions of the hyperbolic tangent and softmax scaled by a factor of 0.1

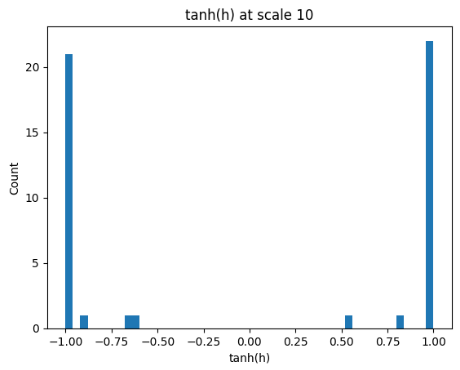


(a) Hyperbolic tangent

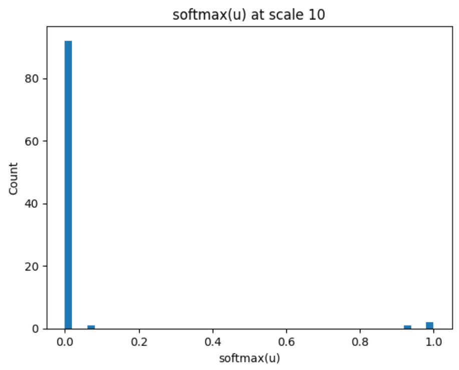


(b) Softmax

Figure 7: Result distributions of the hyperbolic tangent and softmax scaled by a factor of 1

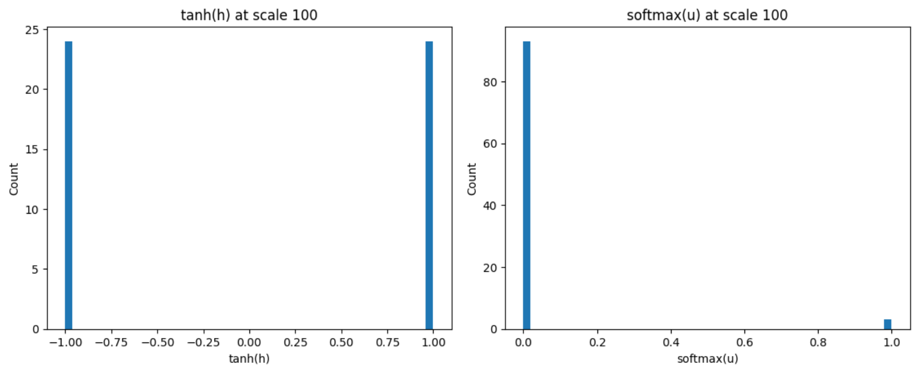


(a) Hyperbolic tangent



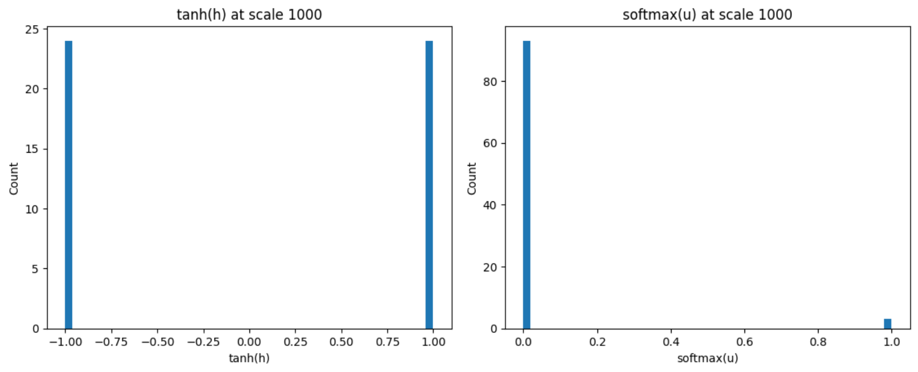
(b) Softmax

Figure 8: Result distributions of the hyperbolic tangent and softmax scaled by a factor of 10



(a) Hyperbolic tangent (b) Softmax

Figure 9: Result distributions of the hyperbolic tangent and softmax scaled by a factor of 100



(a) Hyperbolic tangent (b) Softmax

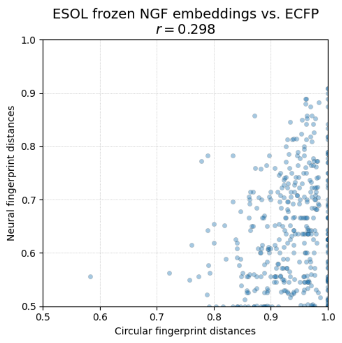
Figure 10: Result distributions of the hyperbolic tangent and softmax scaled by a factor of 1000

This preliminary experiment empirically supports the theoretical foundations that scaling the weights of an NGF aligns the embeddings with ECFP in terms of sparsity.

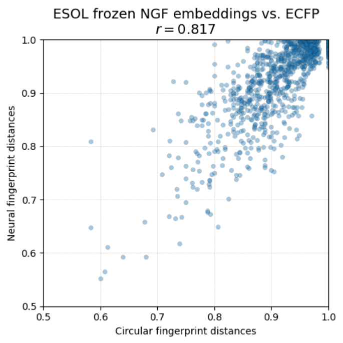
7.4.2 Pairwise distance correlation across weight magnitudes

To investigate how scaling the model's initial weights affects its structural similarity to ECFP, the pairwise distance correlation was measured across several weight magnitudes for the frozen NGF model. Figure 11 presents the results of this experiment on the ESOL dataset [5], comparing the distances to binary ECFP.

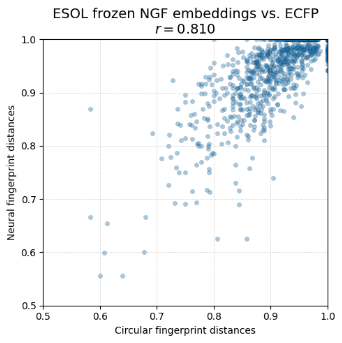
With a small weight scaling factor of 0.1, the correlation is very low ($r = 0.298$), indicating minimal structural alignment. A significant increase is seen when the scaling factor is raised to 1, where the Pearson coefficient jumps to its peak value of $r = 0.817$. However, further increasing the weight magnitude to 10, 100, and 1000 does not yield additional improvements in correlation. The coefficient plateaus, with values of $r = 0.810$, $r = 0.815$, and $r = 0.814$, respectively. This observation shows that the highest structural alignment is achieved at a moderate weight scaling, with larger weights offering no further benefit in this setting. In fact, the highest correlation is achieved with a weight scaling factor of 1, challenging the original paper's claim that using very large weights is the optimal path to mimicking ECFP [6].



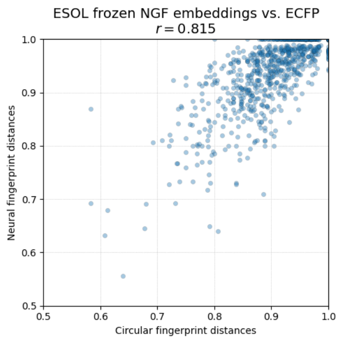
(a) Weight scaling factor: 0.1



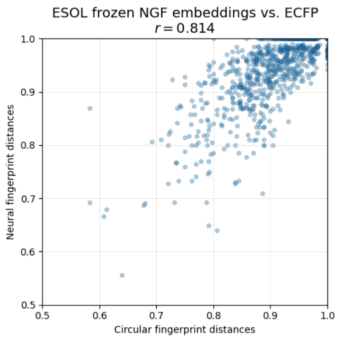
(b) Weight scaling factor: 1



(c) Weight scaling factor: 10



(d) Weight scaling factor: 100



(e) Weight scaling factor: 1000

Figure 11: Effect of weight scaling on the structural correlation of frozen NGF embeddings to Binary ECFP on the ESOL dataset.

7.4.3 Bit Sparsity across weight magnitudes

During these experiments, the bit sparsity of the NGF embeddings was also collected. The results can be seen in table 12 below.

While increasing weight magnitudes does lead to sparser embeddings, this sparsity does not necessarily translate into a better structural alignment with ECFP.

Weight scale	Bit Sparsity
0.1	0.0000
1	0.9035
10	0.9889
100	0.9913
1000	0.9916

Table 12: Bit sparsity of NGF embeddings across weight magnitudes

7.4.4 Frozen downstream task performance across weight magnitudes

Alongside structural similarity, the impact of weight scaling on the frozen embeddings' practical utility for prediction tasks was also evaluated. Table 13 presents the results on the ESOL dataset for each weight magnitude.

Performance is worst at the lowest weight scale of 0.1, which yields a very high RMSE. As the scale increases, the error drops significantly, reaching its optimal (lowest) value of 0.8982 at a weight scale of 10. Beyond this point, further increasing the weight scale is detrimental to performance.

Weight scale	RSME $\downarrow$
0.1	1.6434
1	0.9530
10	**0.8982**
100	0.9766
1000	0.9683

Table 13: Frozen downstream task performance on the ESOL dataset across weight magnitudes

7.5 Adapted NGF Variations

7.5.1 Bond weighted sum

In this experiment, the default sum aggregation was exchanged for a bond weighted sum aggregation. In molecular graphs, bonds carry important information that can directly influence molecular properties. Standard sum aggregation as proposed in the original NGF paper treats all bonds equally. A bond weighted sum aggregation addresses this limitation by explicitly incorporating the bond order as multiplicative weights during neighbourhood aggregation. This approach of edge conditioning aligns with prior work by Yang et al [26] and Simonovsky and Nomodakis [20].

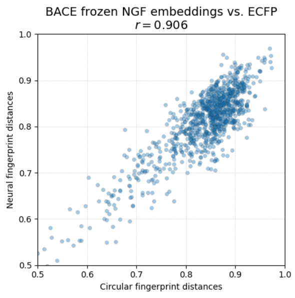
The performance of the frozen embeddings, as shown in table 14, was identical to the baseline NGF, indicating that without training, this change had no effect. However, upon end-to-end training (table 15), a notable improvement was observed on the Lipophilicity dataset, where the RMSE decreased to 0.6370. Performance on the BACE and ESOL datasets remained comparable to the baseline. The structural correlation to Binary ECFP, both frozen and trained, remained mostly similar to the default model (table 27). This suggests that while bond-weighted aggregation can improve predictive performance on specific tasks, it does not alter the model's fundamental structural similarity to ECFP.

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8408
Regression	ESOL	RMSE = 0.9530
Regression	Lipophilicity	RMSE = 0.8128

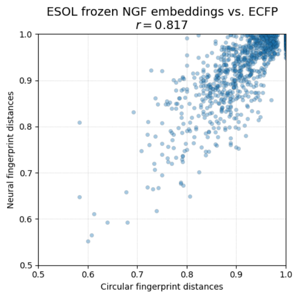
Table 14: NGF with bond weighted sum aggregation frozen performance across datasets

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8541
Regression	ESOL	RMSE = 0.8246
Regression	Lipophilicity	RMSE = 0.6370

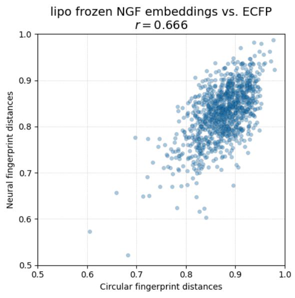
Table 15: NGF with bond weighted sum aggregation end-to-end performance across datasets



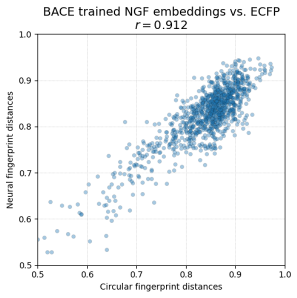
(a) BACE



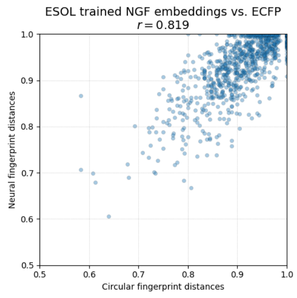
(b) ESOL



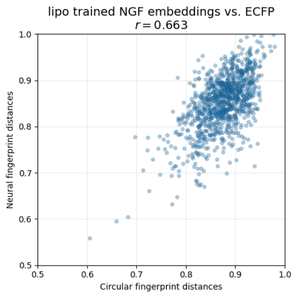
(c) Lipophilicity



(d) BACE



(e) ESOL



(f) Lipophilicity

Figure 12: Correlation of pairwise distances between frozen and trained NGF embeddings vs Binary ECFP

7.5.2 Identity in place of tanh

In this experiment, the tanh smooth function was replaced by the identity function, effectively removing the non-linear activation between layers.

The results show a notable divergence between frozen and trained performance. In the frozen setting, this linear model achieved surprisingly strong results, as detailed in table 16. It outperformed the baseline NGF on both the BACE (0.8702 ROC-AUC) and Lipophilicity (0.7763 RMSE) datasets.

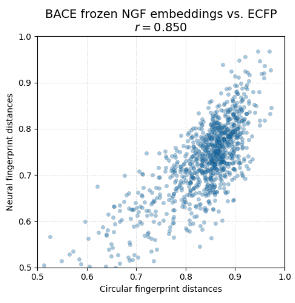
However, this advantage disappeared upon end-to-end training. As seen in table 17, the performance of the trained model was worse than the baseline across all three datasets. Furthermore, removing the non-linearity had a significant negative impact on structural alignment. The correlation to Binary ECFP dropped to $r = 0.850$ on BACE in the frozen state and further degraded to $r = 0.752$ when trained, the lowest correlation observed among the architectural variants (table 27).

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8702
Regression	ESOL	RMSE = 0.9527
Regression	Lipophilicity	RMSE = 0.7763

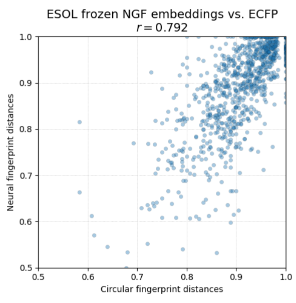
Table 16: Frozen performance NGF with identity function instead of tanh across datasets

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8539
Regression	ESOL	RMSE = 0.9170
Regression	Lipophilicity	RMSE = 0.8351

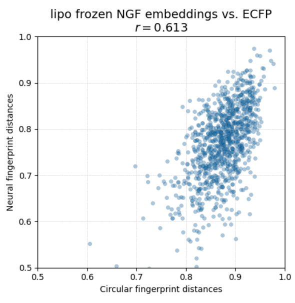
Table 17: NGF with identity function instead of tanh end-to-end performance across datasets



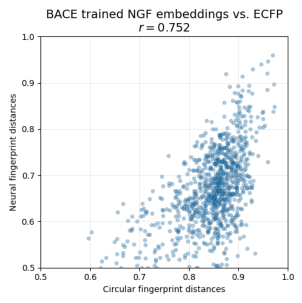
(a) BACE



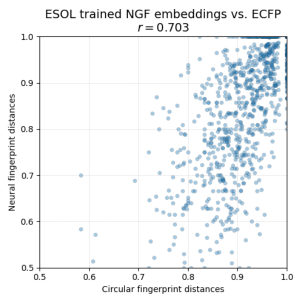
(b) ESOL



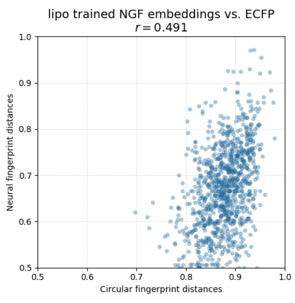
(c) Lipophilicity



(d) BACE



(e) ESOL



(f) Lipophilicity

Figure 13: Correlation of pairwise distances between frozen and trained NGF embeddings vs Binary ECFP

7.5.3 Sine in place of tanh

In this experiment, $\tanh(x)$ was swapped out for $\sin(x)$. This introduces a periodic, non-monotonic transformation. This approach is supported by recent work by Sitzmann et. al, showcasing the surprising effectiveness of using periodic, non-monotonic activation functions [21].

The introduction of the sine function yielded remarkable results in terms of structural alignment. As shown in the summary of correlations (table 27), this variant produced the highest structural similarity to Binary ECFP out of all models tested. In its frozen state, it achieved Pearson coefficients of $r = 0.954$ on BACE, $r = 0.839$ on ESOL and $r = 0.803$ on Lipophilicity. This high degree of similarity was largely retained even after end-to-end training.

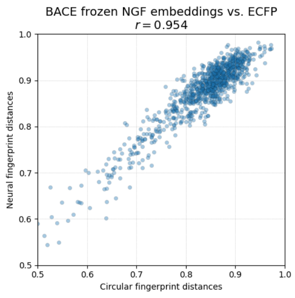
In terms of downstream task performance, the frozen model (table 18) performed comparably to the baseline NGF. When trained end-to-end (table 19), the model achieved a strong ROC-AUC of 0.8753 on the BACE dataset, almost matching the best-performing models. While its performance on the regression tasks was solid, it did not surpass the baseline tanh model. This demonstrates that a periodic activation function can significantly enhance structural similarity to ECFP while maintaining competitive predictive performance.

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8604
Regression	ESOL	RMSE = 0.9459
Regression	Lipophilicity	RMSE = 0.8175

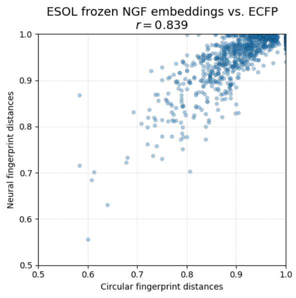
Table 18: Frozen performance of NGF with sine smooth function across datasets

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8753
Regression	ESOL	RMSE = 0.8471
Regression	Lipophilicity	RMSE = 0.6989

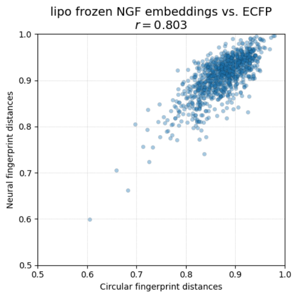
Table 19: End-to-end performance of NGF with sine smooth function across datasets



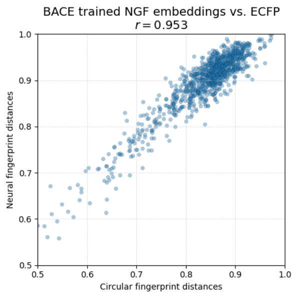
(a) BACE



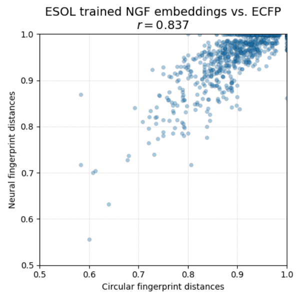
(b) ESOL



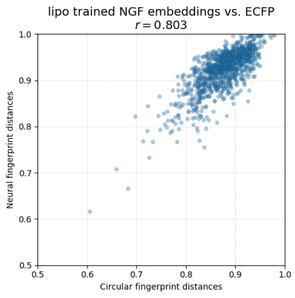
(c) Lipophilicity



(d) BACE



(e) ESOL



(f) Lipophilicity

Figure 14: Correlation of pairwise distances between frozen and trained NGF embeddings vs Binary ECFP

7.5.4 Sparsemax

This experiment explores the use of sparsemax as a sparsify function instead of softmax. In the original NGF architecture, softmax is used to emphasize the most prominent features while surpressing the rest. The softmax function produces dense probability distributions, meaning that all dimensions retain non-zero values, even if they are very small. Sparsemax outputs sparse probability distributions where some dimensions are exactly zero. This makes it more suitable for tasks which benefit from a harder selection or pruning of irrelevant features which is in theory more similar to when ECFP sets a bit in the fingerprint vector [13].

The results indicate that sparsemax successfully promotes sparsity. As shown in table 22, the bit sparsity of the embeddings is high and comparable to that of traditional ECFP. In terms of structural alignment, the sparsemax variant performed well, largely preserving the high correlation of the baseline model in its frozen state (table 27). However, a slight decrease in correlation was observed after end-to-end training.

Regarding downstream task performance, the frozen model (table 20) achieved a solid performances, slightly below the default NGF variant. When trained (table 21), the model's performance was clearly inferior to the default NGF variant. These observations suggest that while sparsemax effectively creates sparse embeddings, this property does not necessarily translate to superior predictive performance compared to the standard softmax function.

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8359
Regression	ESOL	RMSE = 0.9674
Regression	Lipophilicity	RMSE = 0.8273

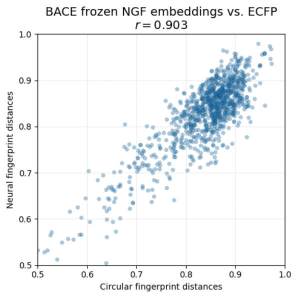
Table 20: Frozen performance of NGF with sparsemax across datasets

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8431
Regression	ESOL	RMSE = 0.8845
Regression	Lipophilicity	RMSE = 0.7226

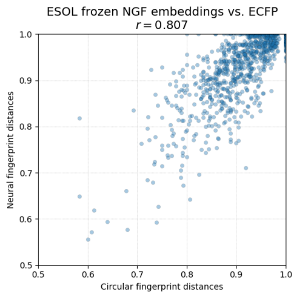
Table 21: End-to-end performance of NGF with sparsemax across datasets

Dataset	**Metric (Value)**
BACE	Bit Sparsity = 0.9711
ESOL	Bit Sparsity = 0.9893
Lipophilicity	Bit Sparsity = 0.9779

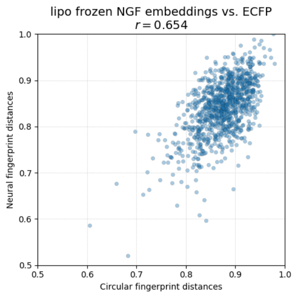
Table 22: Bit Sparsity of NGF embeddings



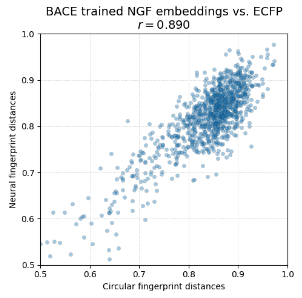
(a) BACE



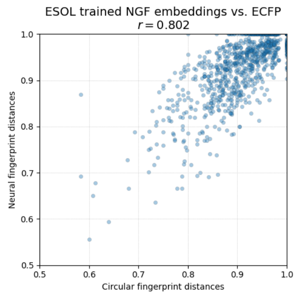
(b) ESOL



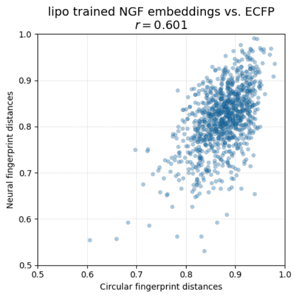
(c) Lipophilicity



(d) BACE



(e) ESOL



(f) Lipophilicity

Figure 15: Correlation of pairwise distances between frozen and trained NGF embeddings vs Binary ECFP

7.5.5 Gumbel-Softmax

In this experiment, softmax was exchanged for Gumbel-Softmax as a sparsify function. Gumbel-Softmax enables models to sample from a discrete distribution while allowing gradients to propagate through the sampling process during training [7, 12]. During this experiment, the temperature parameter of the Gumbel-Softmax function was set to $\tau = 0.5$.

The results for the frozen model showed poor performance across the board. As detailed in table 23, the downstream task performance was significantly worse than the baseline NGF. This was coupled with a very low structural correlation to ECFP (e.g., $r = 0.798$ on BACE), as shown in table 27.

However, the model's performance recovered upon end-to-end training. Table 24 shows that the trained model achieved a strong ROC-AUC of 0.8756 on BACE, making it the top-performing variant. Similarly, the structural correlation to ECFP improved significantly after training, rising to $r = 0.885$ on the BACE dataset. As expected, the bit sparsity of the embeddings was very high, as detailed in table 25. This indicates that while a frozen Gumbel-Softmax model is not effective, it can learn to become a powerful and structurally aligned predictor when trained end-to-end.

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.7708
Regression	ESOL	RMSE = 1.1812
Regression	Lipophilicity	RMSE = 0.9970

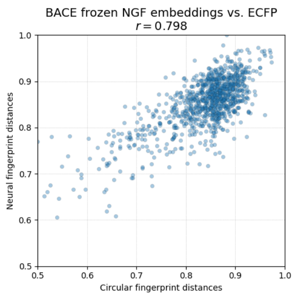
Table 23: Frozen performance of NGF with Gumbel Softmax across datasets

Task	**Dataset**	**Metric (Value)**
Binary Classification	BACE	ROC_AUC = 0.8756
Regression	ESOL	RMSE = 0.8252
Regression	Lipophilicity	RMSE = 0.6689

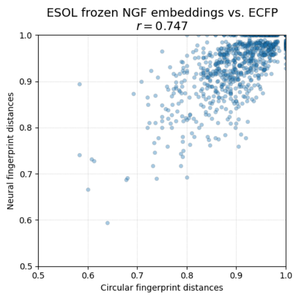
Table 24: End-to-end performance of NGF with Gumbel-Softmax across datasets

Dataset	**Metric (Value)**
BACE	Bit Sparsity = 0.9755
ESOL	Bit Sparsity = 0.9908
Lipophilicity	Bit Sparsity = 0.9813

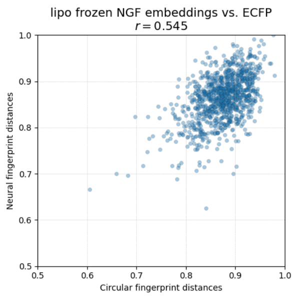
Table 25: Bit Sparsity of NGF embeddings



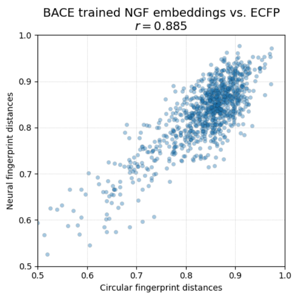
(a) BACE



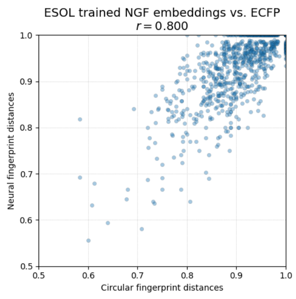
(b) ESOL



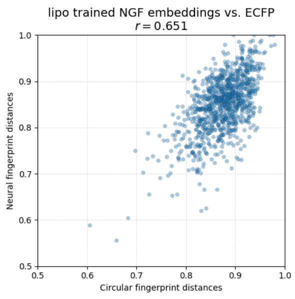
(c) Lipophilicity



(d) BACE



(e) ESOL



(f) Lipophilicity

Figure 16: Correlation of pairwise distances between frozen and trained NGF embeddings vs Binary ECFP

7.6 Summary of Results

7.6.1 Downstream task performance

To provide a comprehensive overview of predictive capabilities, the downstream task performance of all ECFP and NGF variants is summarized in table 26. The table compares the methods based on their ROC-AUC for the BACE classification task and their RMSE for the ESOL and Lipophilicity regression tasks. The best-performing models for each task are highlighted, allowing for a direct comparison of the baseline methods against the architectural adaptations investigated in this thesis.

Method	**ROC_AUC** (BACE) ↑	**RMSE** (ESOL) ↓	**RMSE** (Lipophilicity) ↓
Binary ECFP	0.8646	1.1919	0.8827
Count ECFP	0.8668	0.8704	0.7700
Algorithm 1	0.8484	1.1646	0.8875
Frozen NGF (tanh+softmax)	0.8408	0.9530	0.8128
Trained NGF (tanh+softmax)	0.8627	**0.8009**	0.6671
PyTorch Geometric (frozen)	0.6593	1.7478	1.2369
PyTorch Geometric (trained)	0.8434	0.8122	0.6543
Frozen NGF (bond weighted sum)	0.8408	0.9530	0.8128
Trained NGF (bond weighted sum)	0.8541	0.8246	**0.6370**
Frozen NGF (identity+softmax)	0.8702	0.9527	0.7763
Trained NGF (identity+softmax)	0.8539	0.9170	0.8351
Frozen NGF (sin+softmax)	0.8604	0.9459	0.8175
Trained NGF (sin+softmax)	0.8753	0.8471	0.6989
Frozen NGF (tanh+sparsemax)	0.8359	0.9674	0.8273
Trained NGF (tanh+sparsemax)	0.8431	0.8845	0.7226
Frozen NGF (tanh+gumbel softmax)	0.7708	1.1812	0.9970
Trained NGF (tanh+gumbel softmax)	**0.8756**	0.8252	0.6698

Table 26: Overview of downstream task performance of various fingerprinting techniques across the BACE, ESOL, and Lipophilicity datasets.

7.6.2 Correlation of pairwise distances to Binary ECFP

To evaluate the structural alignment of the different NGF models with traditional fingerprints, their pairwise distance correlation to Binary ECFP was calculated. Table 27 consolidates the Pearson correlation coefficients (r) for all NGF configurations, both in their frozen (randomly initialized) and trained states. This summary allows for a direct assessment of how each architectural choice, impacts the model's ability to mimic the structural relationships captured by ECFP.

Model / Embedding Type	BACE (r)	ESOL (r)	Lipophilicity (r)
Frozen NGF (tanh+softmax)	0.906	0.817	0.666
Trained NGF (tanh+softmax)	0.907	0.821	0.665
PyTorch Geometric (frozen)	0.345	0.240	0.220
PyTorch Geometric (trained)	0.923	0.803	0.713
Frozen NGF (bond weighted sum)	0.906	0.817	0.666
Trained NGF (bond weighted sum)	0.912	0.819	0.663
Frozen NGF (identity+softmax)	0.850	0.792	0.613
Trained NGF (identity+softmax)	0.752	0.703	0.491
Frozen NGF (sin+softmax)	**0.954**	**0.839**	**0.803**
Trained NGF (sin+softmax)	0.953	0.837	**0.803**
Frozen NGF (tanh+sparsemax)	0.903	0.807	0.654
Trained NGF (tanh+sparsemax)	0.890	0.802	0.601
Frozen NGF (tanh+gumbel softmax)	0.798	0.747	0.545
Trained NGF (tanh+gumbel softmax)	0.885	0.800	0.651

Table 27: Structural correlation (Pearson's r) of all NGF variants to Binary ECFP.

8 Discussion

This thesis revisited Neural Graph Fingerprints and their relation to Extended Connectivity fingerprints. It evaluated the NGF algorithm both on an implementational level as well as on its architectural adaptability to assess its structural similarity to ECFP [6, 4]. The results in Tables 26 and 27 provide meaningful insight into the trade-off between structural similarity to ECFP and downstream task performance across different NGF variants and datasets. Among the adapted NGF variants, the sine+softmax variation stands out as having the highest structural similarity to Binary ECFP ($r = 0.954$ on BACE, $r = 0.839$ on ESOL, $r = 0.803$ on Lipophilicity) when frozen and almost retaining or even increasing this similarity when trained ($r = 0.953$ on BACE, $r = 0.837$ on ESOL, $r = 0.803$ on Lipophilicity) while also showcasing solid performance on downstream task. In contrast, the identity+softmax variation showed surprisingly good results in frozen downstream task performance on the BACE and Lipophilicity dataset (0.8702 ROC_AUC, 0.7763 RSME). However, this variation performed worse on these datasets when trained end to end. Sparsification strategies showed mixed results as well. Sparsemax preserved structural correlation but did not outperform the default variant in predictive performance. Gumbel-Softmax on the other hand showed low structural correlation to ECFP and poor predictive performance in the frozen setting but recovered when trained. The bond weighted sum variant showed increased downstream task performance on the Lipophilicity dataset (0.6370 RSME) and equal structural similarity to the default variant. The existing implementation from PyTorch Geometric performed the worst in frozen downstream task performance but recovered significantly when trained, achieving the highest ROC_AUC of 0.8756 on the BACE dataset.

8.1 Critical Analysis of the methodology

Algorithm 1: The algorithmic description of ECFP provided in the works by Duvenaud et. al is a simplified abstraction of official implementations compared to the official ECFP algorithm introduced by David Rogers and Matthew Hahn [4]. In the paper's corresponding codebase however, an official implementation based on RDKit is used. This creates inconsistencies between algorithmic description and practical comparison [6]. This is supported by the results in table 4.

Large random weights: The use of large random weights was repeatedly mentioned in the paper by Duvenaud et. al. However, the magnitude of those weights were not defined quantitatively. Furthermore, the empirical findings of this thesis suggest that using large random weights only aligns NGF embeddings and ECFP in terms of sparsity (tables:2, 12) while actually being mostly detrimental to pairwise distance correlation (figure: 11) and downstream task performance (table: 13) [6]. This suggests a potential flaw in the analogy that the hyperbolic tangent acts as a soft hash in the limit of large weights. While large weights make tanh behave like a step function, they also cause a significant loss of information by not distinguishing between values that are far from zero. This information loss may even be more detrimental for the structural representation than the benefit of sparsity alignment with ECFP.

Similarity to Count ECFP: The observation that NGF embeddings are more similar to Count ECFP (figure 5) can be explained by the bit accumulation strategy used in the NGF algorithm. Each sparsified vector gets added to the final fingerprint. Since the method does not take any measures to deduplicate or binarize the final molecular embedding, it makes total sense that it is more similar to Count ECFP. Since Count ECFP is performing consistently better than Binary ECFP across downstream tasks, this similarity may actually be desirable.

NGF Adaptations: In this thesis, several adaptations of the NGF architecture were tested. The use of the periodic sine function yielded a higher structural alignment of NGF embeddings and Binary ECFP than the use of the hyperbolic tangent. A possible explanation for this could be that the sine function avoids the information saturation which is inherent in the hyperbolic tangent. With tanh inputs that are far away from 0 are mapped to either 1 or -1. The sine function on the other hand retains distinctions even for inputs with a large absolute value. This potentially leads to more accurate distinction of neighbourhoods that the hyperbolic tangent would collapse into the same value, leading to a more nuanced embedding. Other adaptations, such as bond weighted neighbourhood aggregation, Sparsemax and Gumbel-Softmax showed benefits in some cases but did not consistently outperform the baseline NGF architecture although both Sparsemax and Gumbel-Softmax promoted sparsity similar to ECFP (tables 25, 22, 2) without the need for weight scaling. This further supports the finding that sparsity similar to ECFP does not necessarily promote structural similarity and downstream task performance. Overall, no single NGF variant dominates the others in terms of structural alignment to ECFP and downstream task performance. All experiments in this thesis were run with multiple random seeds to investigate the robustness of the findings. While the relative ranking in terms of pairwise distance correlation to ECFP was mostly preserved, downstream task performance varied significantly. This reflects the stochastic nature of random weight initialization. These explorative experiments and results should be seen as preliminary. To draw precise conclusions, further experimentation is needed.

8.2 Existing implementation in PyTorch Geometric

An existing implementaion of Neural Graph Fingerprints is available in PyTorch Geometric [16]. This implementation failed to reproduce key findings like the plot in figure 3 in the paper by Duvenaud et. al [6]. This suggests that this implementation deviates from the algorithmic description on the original paper.

9 Conclusions and Future Work

This goal of this thesis was to evaluate how well NGF embeddings approximate ECFP structurally and functionally. NGF embeddings have shown to be more similar to Count ECFP than to Binary ECFP which is a positive outcome considering that Count ECFP outperforms Binary ECFP in downstream tasks. The algorithmic description of ECFP in the NGF paper has shown to be significantly different to official ECFP implementations. Exchanging the hyperbolic tangent function for the periodic sine function improved correlation in pairwise distances to ECFP.

To ensure statistical robustness, the experiments that were conducted in this thesis should be repeated on more diverse datasets. In all the experiments of this thesis the atom features used to generate NGF embeddings were restricted by the atom features that were used in the paper by Duvenaud et. al [6] to ensure comparability to the original findings. Future work could repeat these experiments with broader atom features to investigate the capabilities of Neural Graph Fingerprints more intensively. Additionally, using distance metrics other than the continuous generalization of the Tanimoto distance could be an interesting approach to further investigate the similarity of NGF embeddings to ECFP. Furthermore, future work could analyze the effect of choosing different hyperparameters like fingerprint dimension, ECFP radius or the number of

NGF layers. Exchanging the hyperbolic tangent for the periodic sine function exhibited improvements in structural alignment of NGF embeddings to ECFP. Further experiments could investigate the use of other periodic and non monotonic functions.

10 Acknowledgements

I want to express my gratitude to Anatol Ehrlich and Prof. Nils Kriege for providing me with the opportunity to work on this thesis. They provided valuable feedback as well as guidance throughout the process of conducting this study and writing this report. Furthermore, I want to express my deepest gratitude to my dear friends Kamil Kruk and Kerim Atak not only for their valuable feedback on this report, but also for the countless intellectual exchanges and discussions during our studies.

11 Appendix

All the code as well as the raw results to this work is available on GitHub: `https://github.com/albertd01/BSC_thesis`

References

[1] Amézqueta, S., Subirats, X., Fuguet, E., Rosés, M., and Ràfols, C. Chapter 6 - octanol-water partition constant. In *Liquid-Phase Extraction*, C. F. Poole, Ed., Handbooks in Separation Science. Elsevier, 2020, pp. 183–208.

[2] Capecchi, A., Probst, D., and Reymond, J.-L. One molecular fingerprint to rule them all: Drugs, biomolecules, and the metabolome. *Journal of Cheminformatics 12* (06 2020).

[3] Corso, G., Cavalleri, L., Beaini, D., Liò, P., and Veličković, P. Principal neighbourhood aggregation for graph nets, 2020.

[4] David Rogers, M. H. Extended connectivity fingerprints. *J. Chem. Inf. Model. 2010, 50, 742-754* (2010).

[5] Delaney, J. S. ESOL: Estimating Aqueous Solubility Directly from Molecular Structure. *Journal of Chemical Information and Computer Sciences* (4 2019).

[6] Duvenaud, D., Maclaurin, D., Aguilera-Iparraguirre, J., Gómez-Bombarelli, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. Convolutional networks on graphs for learning molecular fingerprints, 2015.

[7] Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax, 2017.

[8] Kearnes, S., McCloskey, K., Berndl, M., Pande, V., and Riley, P. Molecular graph convolutions: moving beyond fingerprints. *Journal of Computer-Aided Molecular Design 30*, 8 (Aug. 2016), 595–608.

[9] keiserlab. keras-neural-graph-fingerprint: Keras implementation of neural graph fingerprints. `https://github.com/keiserlab/keras-neural-graph-fingerprint`, 2017. Commit `cab0edb` (May 17, 2017); accessed: 2025-06-26.

[10] Keyulu Xu, Weihua Hu, J. L., and Jegelka, S. How powerful are graph neural networks. *Conference paper at ICLR 2019* (2019).

[11] Landrum, G., and the RDKit Contributors. *The RDKit Documentation*, 2025.3.3 ed. RDKit Open-Source Community, 2025. Accessed: 2025-06-26.

[12] Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables, 2017.

[13] Martins, A. F. T., and Astudillo, R. F. From softmax to sparsemax: A sparse model of attention and multi-label classification, 2016.

[14] Polgar, T., Magyar, C., Simon, I., and Keseru, G. Impact of ligand protonation on virtual screening against beta-secretase (bace1). *Journal of chemical information and modeling 47* (11 2007), 2366–73.

[15] PyTorch Geometric Contributors. *PyTorch Geometric Documentation*, 2.6.1 ed. PyTorch Geometric, 2025. Accessed: 2025-06-26.

[16] PyTorch Geometric Contributors. *PyTorch Geometric Neural Fingerprint*, 2.5.2 ed. PyTorch Geometric, 2025. Accessed: 2025-06-26.

[17] Riniker, S., and Landrum, G. Open-source platform to benchmark fingerprints for ligand-based virtual screening. *Journal of cheminformatics 5* (05 2013), 26.

[18] Saraiva, P. On shannon entropy and its applications. *Kuwait Journal of Science 50*, 3 (2023), 194–199.

[19] Schütt, K. T., Kindermans, P.-J., Sauceda, H. E., Chmiela, S., Tkatchenko, A., and Müller, K.-R. Schnet: A continuous-filter convolutional neural network for modeling quantum interactions, 2017.

[20] Simonovsky, M., and Komodakis, N. Dynamic edge-conditioned filters in convolutional neural networks on graphs. *CoRR abs/1704.02901* (2017).

[21] Sitzmann, V., Martel, J. N. P., Bergman, A. W., Lindell, D. B., and Wetzstein, G. Implicit neural representations with periodic activation functions, 2020.

[22] Song, Y., Zheng, S., Niu, Z., Fu, Z.-H., Lu, Y., and Yang, Y. Communicative representation learning on attributed molecular graphs. 2803–2810.

[23] Wang, J.-B., Cao, D.-S., Zhu, M.-F., Yun, Y.-H., Xiao, N., and Liang, Y.-Z. *In silico* evaluation of $\log D_{7.4}$ and comparison with other prediction methods. *Journal of Chemometrics 29*, 7 (2015), 389–398.

[24] Weininger, D. Smiles, a chemical language and information system. 1. introduction to methodology and encoding rules. *American Chemical Society* (1987).

[25] Wu, Z., Ramsundar, B., Feinberg, E., Gomes, J., Geniesse, C., Pappu, A. S., Leswing, K., and Pande, V. Moleculenet: a benchmark for molecular machine learning. *Chem. Sci. 9* (2018), 513–530.

[26] Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., Guzman-Perez, A., Hopper, T., Kelley, B., Mathea, M., Palmer, A., Settels, V., Jaakkola, T., Jensen, K., and Barzilay, R. Analyzing learned molecular representations for property prediction, 2019.