THE UNIVERSITY OF SYDNEY

The University of Sydney

School of Pharmacy

AmesFormer - A Graph Transformer Neural Network for Mutagenicity Prediction

Author:

Luke Thompson

Supervisor:

Dr. Slade Matthews

A thesis submitted for the completion of:

Bachelor of Science (Honours)

July 22, 2024

9522 words

Contents

0.1 Acknowledgements

I would like to thank Dr. Slade Matthews, my honours supervisor, for his guidance on this project. After my late transfer into the Computational Pharmacology & Toxicology Lab (CPT Lab), Dr. Matthews ensured I was never lost and always had a clear direction toward completing this project on time. Our many one-on-ones in the office and outside provided much insight, encouragement and humour. His support and expertise have been crucial to my success and growth throughout this journey.

I also thank Dr. Matthews and Dr. Davy Guan, a lab graduate and current CSIRO scientist, for their efforts to secure me an industry partnership for the Ph.D I hope to pursue.

I would also like to thank Professor Renae Ryan AM who graciously supported my transfer from her laboratory to the CPT Lab. Whilst I did not stay in the transporter biology group, this time helped me realise where my research passions truly lay - for that I am very grateful.

Thank you to Josiah Evans, a friend of over a decade and programmer-extraordinaire, who provided invaluable guidance throughout the final weeks of my project. For those late nights helping me fix

off-by-one tensor product bugs and many tangential discussions about current research in machine learning I am especially grateful.

I would like to thank my partner Miriam who has supported me throughout my honours, especially for her wonderful cooking the many nights I was busy writing this thesis. Thank you also to my parents who have supported me all the way through my undergraduate.

Finally, thank you to Dewy, Xin and Daniella of the CPT Lab for enduring my weekly lab meeting whiteboard sessions. Their many questions, suggestions and sometimes confused looks pressed me to refine my own understanding of my research.

0.2 Declaration

Declaration/Compliance statement:

1. I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure;

2. I understand that failure to comply with the Student Plagiarism Coursework Policy and Procedure can lead to the University commencing proceedings against me/us for potential student misconduct under Chapter 8 of the University of Sydney By-Law 1999 (as amended);

3. This work is substantially my own, and to the extent that any part of this work is not my own, I have indicated that it is not my own by acknowledging the source of that part of the work.

Table 1: Declaration of work.

Name	**Contributions**
Luke Thompson	Project design, methods design, result gathering and analysis, programming
Slade Matthews	Supervisor, consultation, statistics design
Josiah Evans	Programming assistance regarding performance optimisation
ChatGPT 4.0	Generation of Matplotlib code for results section height

0.3 Abstract

The Ames mutagenicity test is a gold standard assay for the safety assessment of all new chemical entities and represents a high bar to market access for pharmaceuticals. However, as the Ames test is prohibitively expensive to run for all new chemicals, several quantitative structure activity relationship (QSAR) models of Ames mutagenicity have been proposed. Despite their high performance, the best performing models rely on challenging-to-interpret ensemble strategies and molecular fingerprint data which often neglects gestalt molecular structure. To improve upon these models, we propose AmesFormer , a graph transformer network (GTN) which shows state-of-the-art single model performance for the prediction of Ames mutagenicity. We first review current literature surrounding the computational assessment of Ames mutagenicity, including a formal discussion of neural network (NN) and graph neural network (GNN) architectures. We discuss the architecture of AmesFormer with specific reference to the generalisation of transformer positional encodings to chemical data and related encodings. We benchmark AmesFormer using a standardised dataset against 22 other Ames models, achieving best-in-class balanced accuracy and F1 score among non-ensemble models. We perform exploratory clustering and principal components analysis (PCA) on our datasets, and quantify the uncertainty of our performance metrics using a unique Bayesian approach. Additionally, we provide empirical support for our model's performance using other GTN benchmarks from the literature and complement this with theoretical developments in discrete mathematics and graph theory. Overall, we present a high-performance, accessible, and open-source computational model for Ames mutagenicity, with significant potential for regulatory and drug development applications. All code is available on GitHub (https://github.com/leftwinglow/ames_graphormer).

Chapter 1

Introduction

1.1 The Ames Assay

The Ames test is a widely-used *in vitro* mutagenicity assay essential to drug development and regulatory pipelines. It represents a high bar to market access for pharmaceuticals, with many compounds rejected early in the case of an Ames-positive outcome (Honma et al. 2019). The Ames assay is explicitly required by International Council for Harmonisation (ICH) guideline S2 (R1) (ICH 2013).

In the Ames assay, a histidine-deficient substrate is inoculated with strain of auxotrophic mutant histidine-dependent Salmonella typhimirium (Ames et al. 1973). This dependence is introduced via a mutation at the histidine operon (Ames et al. 1973). A suspected mutagenic compound is then introduced. If the mutagen-containing substrate shows significantly greater colony growth than a control, the test-molecule has reversed the histidine-dependence mutation and is mutagenic. This simplicity has resulted in the Ames test having an excellent inter-laboratory replicability of 85% (Kamber et al. 2009).

Different *S. typhimurium* strains detect various mutagenicity mechanisms, which can be categorized into substitution mutations (SNPs) or frameshift mutations (Lui et al. 2023). The Ames test also includes S9 rodent liver homogenate, enabling the detection of mutagens that require metabolic activation (Maron and Ames 1983; Ames et al. 1973).

The Ames test is relatively inexpensive compared to other mutagenicity assays, costing approximately $2000 per chemical. However, as the CAS reigstry grows by over 4000 molecules daily, and with Australia representing a small regulatory environment, the cost of screening every new chemical quickly becomes prohibitive (Honma et al. 2019).

To address this problem, *in silico* quantitative structure activity relationship (QSAR) models have been developed to provide cheaper and higher-throughput methods of Ames screening (Furuhama et al.

2023). The use of such models is recommended within ICH guideline M7 (R1) for the control of mutagenic impurities in pharmaceuticals and the European Union's Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH) agreement (European Communities 2006; ICH 2017; Honma et al. 2019). The Australian Industrial Chemicals Introduction Scheme (AICIS) and the United States Food and Drug Administration (US FDA) have also provided guidelines for implementing *in silico* or other non-animal safety evaluation methods (AICIS 2022; Han 2023).

The Ames test is widely used throughout drug development, food safety, skin sensitisation studies, and regulatory toxicology (Ashby et al. 1993; Patlewicz et al. 2010). *In silico* Ames testing may also be employed as part of the Integrated Approaches to Testing and Assessment (IATA) approach to skin toxicity assessment, directly reducing the need for animals in skin sensitisation studies via multi-assay triangulation (OECD 2020). This research is thus situated within the global effort to enhance the usefulness and robustness of *in silico* Ames testing, and any improvements in the accuracy of *in silico* Ames models can directly reduce regulatory burden and improve public safety.

1.2 Molecular Fingerprints

Most popular QSAR models leverage molecular fingerprints (MFs), which are vectorised representations of a molecule's chemical features (Capecchi et al. 2020). As a type of chemical descriptor, MFs serve as primary inputs to QSAR models of Ames mutagenicity. Two well-established fingerprinting methodologies have been applied to molecules within the Lipinski Limits.

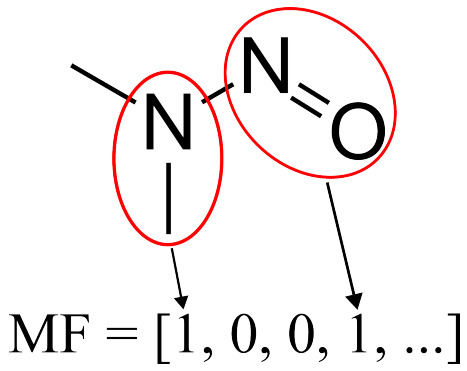


Figure 1.1: An example fingerprint showing the hashing of different NDMA chemical substructrures into the MF bit vector.

1. Structural key MFs describe molecules by encoding the presence or absence of predefined chemical substructures into a unique index within binary bit vector, as shown in Figure 1.1 (Zhong and Guan 2023). The Molecular ACCess Systems (MACCS) key MF, developed by Durant et al. (2002), is the archetypal and most widely used key-type MF for QSAR tasks. MACCS keys, as shown

in Figure 1.2a encode a set of 166 possible substructures into a 166-bit vector (Durant et al. 2002; Seo et al. 2020). This enables straightforward model interpretability as each key corresponds to a known substructure. For example, index 145 in MACCS key MF always denotes the presence of ≥1 six-membered ring (Landrum 2012). Key MFs also allow straightforward indexing of molecule databases to enumerate the presence or absence of specific substructures (Bolton et al. 2008). To this end, PubChem developed PubChemFP, a key MF optimised for compound search and similarity scoring (Seo et al. 2020; Bolton et al. 2008). A side effect of substructure preselection is that key MFs are unable to represent novel moieties outside the predefined set, and therefore underperform on datasets containing novel chemical structures (Zhong and Guan 2023).

$$\text{MACCS} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_{166} \end{bmatrix} \rightarrow \{0,1\}^{166} \qquad \text{ECFP} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \{0,1\}^{n} \qquad \text{ECFP-Count} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \rightarrow \{\mathbb{R}\}^{n}$$

(a) MACCS fingerprint (b) Morgan fingerprint (c) Morgan-count fingerprint

Figure 1.2: Different molecular fingerprinting techniques and their dimensionalities. **a.** MACCS are a bit vector of length 166, encoding the presence (1) or absence (0) of predefined molecular substructures. **b.** Extended-connectivity Fingerprint 4 (ECFP-4) (Morgan FP) encode non-predefined molecular substructures into a bit vector of variable length n. **c.** Morgan FP-count is akin to Morgan FP but the bit vector may take any real-valued scalar $\mathbb{R}$ to encode the *frequency* of each non-predefined substructure.

2. Hash MF escape this problem by *de novo* numerically encoding non-predefined substructures around each constituent atom, in essence encoding each atom's local environment into a bit vector (Rogers and Hahn 2010). Customisable parameters govern the n-hop radius from the central atom of a substructure to describe (the ''size'' of the atom's local environment), and the bit vector length (the maximum number of substructures described within the fingerprint) (Zhong and Guan 2023; Rogers and Hahn 2010). Extended-connectivity Fingerprint 4 (ECFP-4) or Morgan fingerprints, as shown in Figure 1.2b, have become the standard MF modality for small molecule QSAR studies (Capecchi et al. 2020). Zhong and Guan (2023) further develop Morgan FP by encoding the quantity of each substructure into the bit vector, Figure 1.2c, marginally improving QSAR performance. It is also possible to combine multiple fingerprints via concatenation, though this is not common practice (Noutahi et al. 2023).

Hash MF suffer from bit collision, wherein structurally similar chemical fragments are hashed into the same index of the bit vector (Luchini 2021). As the probability of bit collision is inversely proportional to bit vector length, hash MF are often used with lengths of 2048 or 4096. Increasing bit vector length reduces collision frequency by expanding the number of indexes into which a substructure can be placed. However, greater lengths inevitably increase vector sparsity and dimensionality, thus increasing the computational cost of training and potentially hindering model robustness (Wen et al. 2016; Altman and Krzywinski 2018). Thus finding methods to move beyond MFs represents a valuable research direction.

This will be addressed in this thesis with a graph transformer network (GTN) model. In order to understand the development of GTNs, it is useful to first introduce neural networks (NNs) as they were first developed.

1.3 Neural Networks

A NN is a machine learning architecture wherein neurons are arranged into discrete layers, and neurons from one layer are connected to neurons of another layer. The goal of NN training is to produce a model that, when given a set of inputs $\vec{x}$, can accurately predict an associated variable y.

The training of a neural network may be divided into two distinct phases: *feed forward* and *backward propagation*.

Feed Forward. We may consider the forward pass f of a neural network as a sequence of D mappings or *layers*, denoted as $f = f^{(D)} \circ f^{(D-1)} \circ \ldots \circ f^{(1)}$, where each $f^{(d)}$ represents a distinct layer within the network. A layer $f^{(d)}$ takes the output of the previous layer $\vec{x}^{(d-1)}$ as input, and transforms it according to a randomly initialised set of parameters w_d. We may denote this as:

$$x^{(d)} = f^{(d)}(x^{(d-1)}; w_d) \tag{1.1}$$

The parameters w_d of all the layers $f^{(D)}_{w_D} \circ f^{(D-1)}_{w_{D-1}} \circ \ldots \circ f^{(1)}_{w_1}$, are initialised by randomly sampling from a uniform U or normal $\mathcal{N}$ distribution (Glorot et al. 2010; He et al. 2015). Two prevalent initializations are He and Xavier:

$$w_d \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right) \qquad \text{(He initialisation)} \tag{1.2}$$

$$w_d \sim U\left(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}}\right) \qquad \text{(Xavier initialisation)} \tag{1.3}$$

where:

- n_{in} is the number of input neurons, used in He initialization to maintain activation variance (He et al. 2015).
- n represents the average number of input and output neurons (n_{in} and n_{out}) per layer, used in Xavier initialization to balance the variance in forward and backward propagation (Glorot et al. 2010).

During feed forward, each layer in the network performs an affine operation on its inputs. Specifically, when the input to layer $f^{(d)}$ is the vector $\vec{x}^{(d-1)}$, this operation involves taking the dot product of $\vec{x}^{(d-1)}$ with the weight matrix w_d and then adding a bias term b_d, as represented in eq. (1.4). The result of this operation is the unactivated neuron output denoted as $z^{(d)}$. This intermediate result $z^{(d)}$ is processed by the activation function σ, which in the case of He-initialized neural networks is typically the ReLU function shown in Equation (1.5) (He et al. 2015). The output of the ReLU function, $\sigma(z^{(d)})$, becomes the final output of the layer, $\vec{x}^{(d)}$, detailed in Equation (1.6).

$$z^{(d)} = f^{(d)}(\vec{x}^{(d-1)}; w_d, b_d) = w_d\vec{x}^{(d-1)} + b_d \tag{1.4}$$

$$\sigma(z^{(d)}) = \max(0, z^{(d)}) \tag{1.5}$$

$$\vec{x}^{(d)} = \sigma(z^{(d)}) \tag{1.6}$$

In essence, during *feed forward* operation, the value of each input to a neuron is multiplied by a weight matrix, added to a bias, and summed. This sum is passed through an *activation function*, then output to be multiplied by the next weights connecting the neuron to neurons of subsequent layers. The output of the final layer, $x^{(D)}$, therefore represents the network's prediction $\hat{y}$ given $\vec{x}$.

Backward propagation. In the case of binary classification, such as determining whether a molecule is Ames positive or negative, the output of the final layer $x^{(D)}$ is a single *logit* representing the log-odds of the probability that the input molecule $\vec{x}$ belongs to the positive class. To obtain the raw probability that $\vec{x}$ belongs to the positive class, the logit is passed through a sigmoid function $P(y = 1|x) = \sigma(\text{logit})$.

All NNs implement a loss function, which quantifies the deviation between $\hat{y}$, the model's prediction, and y, the actual class label used in training. In the case of binary classification, like determining Ames mutagenicity, a binary cross-entropy loss function may be used. This loss function L, which outputs a scalar loss value, $\mathscr{L}$, is defined as:

$$L(y, \hat{y}) = -\left(y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right) = \mathscr{L} \tag{1.7}$$

As the parameters of all layers $f = f_{w_D}^{(D)} \circ f_{w_{D-1}}^{(D-1)} \circ \ldots \circ f_{w_1}^{(1)}$ are randomly initialised, the goal of backpropagation is to to minimise the network's $\mathscr{L}$ by adjusting its parameters via gradient descent (Rumelhart et al. 1986).

First, we find the derivative of the loss function L with respect to the network's final output $\hat{y}$:

$$\frac{\partial L}{\partial \hat{y}} = -\left(\frac{y}{\hat{y}} - \frac{1-y}{1-\hat{y}}\right) \tag{1.8}$$

We then backpropagate this gradient $\frac{\partial L}{\partial \hat{y}}$ from the output layer D to the input layer d. By employing the chain rule, we can then determine the gradient of L with respect to every parameter of the model through the product of partial derivatives tracing back from $\hat{y}$ to w. The gradient for each layer d with respect to its weights w_d is thus computed as:

$$\frac{\partial L}{\partial w_d} = \frac{\partial L}{\partial \hat{y}} \cdot \frac{\partial \hat{y}}{\partial \vec{x}^{(D)}} \cdot \prod_{k=D}^{d+1} \left(\frac{\partial \vec{x}^{(k)}}{\partial \vec{x}^{(k-1)}}\right) \cdot \frac{\partial \vec{x}^{(d+1)}}{\partial z^{(d)}} \cdot \frac{\partial z^{(d)}}{\partial w_d} \tag{1.9}$$

Where:

- $\frac{\partial L}{\partial w_d}$ is the derivative of the loss function L with respect to the parameters of each layer d
- $\frac{\partial L}{\partial \hat{y}}$ is the derivative of the loss function with respect to the output of the network $\hat{y}$.
- $\frac{\partial \hat{y}}{\partial \vec{x}^{(D)}}$ is the derivative of the output $\hat{y}$ with respect to the activated output of the last layer $\vec{x}^{(D)}$.
- $\prod_{k=D}^{d+1} \left(\frac{\partial \vec{x}^{(k)}}{\partial \vec{x}^{(k-1)}}\right)$ is the chain of derivatives through each layer from D back to $d+1$, including the effects of the activation functions and the affine transformations between layers.
- $\frac{\partial \vec{x}^{(d+1)}}{\partial z^{(d)}}$ is the derivative of the activated output of layer d with respect to the unactivated output $z^{(d)}$.
- $\frac{\partial z^{(d)}}{\partial w_d}$ is the derivative of the unactviated output $z^{(d)}$ with respect to the parameters w_d at layer d.

These derivatives, such as $\frac{\partial L}{\partial w_d}$, encapsulate how changes in the parameters, like w_d, affect the network's final output, $\hat{y}$. Using gradient descent, we iteratively adjust the network's parameters in the opposite direction of the gradient of the loss function, $\frac{\partial L}{\partial w_d}$. By moving against the gradient, the process descends towards a local minimum on the loss function's surface, which is typically convex, as shown in Figure 1.3.

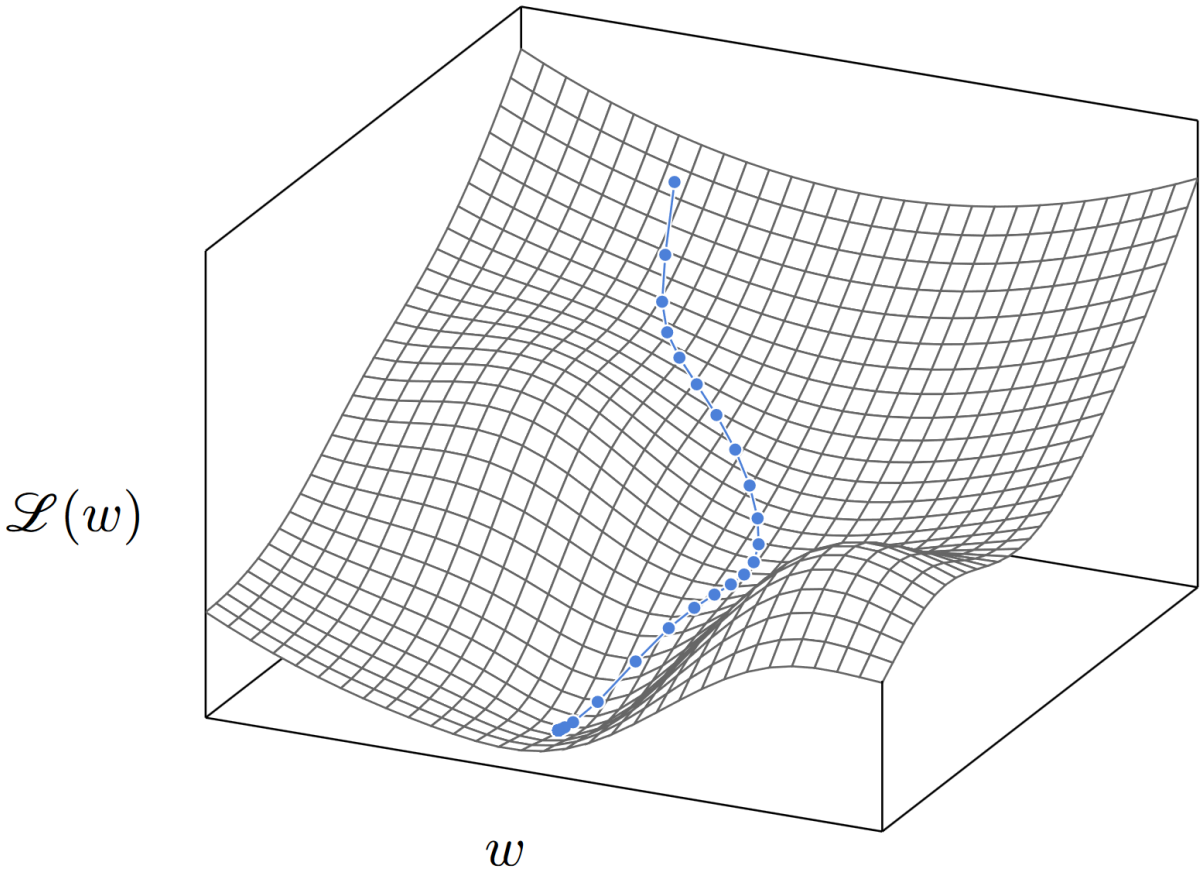


Figure 1.3: Iterative minimisation of the loss function via gradient descent down the convex loss surface. The effect of a linear decay learning rate (LR) scheduler can be seen through the decreasing step-size as the loss progresses closer to a local minimum. Figure from François Fleuret (2023).

The learning rate (LR), denoted η, is a hyperparameter which controls the size of the steps taken towards the local minimum during the optimization process. It may be written as:

$$w_d = w_d - \eta \cdot \frac{\partial L}{\partial w_d} \tag{1.10}$$

Where w_d represents the weights, η is the LR, and $\frac{\partial L}{\partial w_d}$ is the gradient of the loss function with respect to a given weight. The LR is usually adjusted throughout training according to a LR scheduler. A common technique is linear decay, where the LR is decreased over the course of training. In the beginning whilst the LR is high, the model may explore a large region of the loss landscape, but as the LR decreases, the model converges on a local minimum; this may be seen in Figure 1.3.

Hence, a neural network's output $\hat{y}$ begins uncorrelated with input data $\vec{x}$ but iteratively learns the correct output for a given input via minimisation of the loss function L over the course of training (Rumelhart et al. 1986).

During inference, when the model is applied to new data for prediction, the loss value becomes irrelevant as only feed-forward is used. In this case, if $P(y = 1|x)$ exceeds a given threshold, typically 0.5, we classify the molecule as Ames positive; if it is below 0.5, we classify the molecule as Ames negative. This is how a trained model would behave in the regulatory context.

1.4 Transformers

The transformer introduced by Vaswani et al. (2017), has produced a major leap in the capability of machine learning (ML) models across natural language processing (NLP), image processing and biomedical tasks (Devlin et al. 2018; Brown et al. 2020; Parmar et al. 2018; Radford et al. 2021; Elnaggar et al. 2020). An in-depth exploration of the transformer is beyond the scope of this thesis, however, interested readers may consult an excellent review by Lin et al. (2021).

We limit our discussion to the multi-headed scaled dot-product attention blocks because they are largely responsible for the excellent performance of transformers and are implemented within AmesFormer. Formally, given a vector of features $\vec{h}$, we compute the query, key, and value matrices by performing matrix multiplication with three learnable, randomly initialized weight matrices W_Q, W_K, and W_V:

$$Q = \vec{h}W_Q \qquad \text{(shape: } (\vec{h}, d_k)) \tag{1.11}$$

$$K = \vec{h}W_K \qquad \text{(shape: } (\vec{h}, d_k)) \tag{1.12}$$

$$V = \vec{h}W_V \qquad \text{(shape: } (\vec{h}, d_v)) \tag{1.13}$$

Intuitively, we may think of the columns of the query matrix being each feature asking ‘‘what information do I want’’; whilst the key matrix is each feature stating ‘‘this is the information I can give’’. The value vector then reintroduces the actual information to be attended.

We calculate the attention scores by taking the dot product of Q with K^T. To improve numerical stability these scores are scaled by the square root of the dimensionality of the key vectors, denoted as d_k (Vaswani et al. 2017). The scaled attention scores are then passed through a softmax function to obtain a probability distribution over the input features:

$$A = \frac{QK^T}{\sqrt{d_k}} \qquad \text{(shape: } (\vec{h}, \vec{h})) \tag{1.14}$$

This softmax operation ensures that the resulting attention weights sum to one, allowing for a weighted aggregation of the value vectors. The final output of the attention mechanism is obtained

by performing a weighted sum of the value vectors V, where the weights are given by the computed attention scores:

$$\text{Attention}(Q, K, V) = \text{softmax}(A)V \qquad (\text{shape: } (\vec{h}, d_v)) \tag{1.15}$$

The V matrix is adding the actual data back, so it may be attended according to the attention scores. The attention score matrix A has the shape $(|\vec{h}| \times |\vec{h}|)$ and contains these coefficients describing how much each feature should attend to all other features. Specifically, each element A_{ij} represents the attention weight of the i-th feature towards the j-th feature. This allows the model to dynamically adjust the importance of different features when computing the final representation, which has the shape $(|\vec{h}| \times d_v)$. Multiple attention calculations or ''heads'' are performed in parallel on the same input vector using different Q, K and V matrices. Their outputs are later concatenated to form the final representation. This concatenated output is fed through a conventional neural network (NN), identical to those described in the previous section.

In essence, we are computing how much each feature should *attend* to all the others, facilitating the capture of complex interactions and long-range dependencies within the input sequence or how much each molecular substructure may influence the others, in the context of a Morgan fingerprint.

1.5 An Archetypal Neural Network for Cheminformatics

We may consider a simple Ames NN, shown in Figure 1.4, as passing a molecular fingerprint (MF) bit vector as the input to a network similar those described above, which classifies the input as Ames-positive or Ames-negative (Xu, Cheng, et al. 2012). Examples of this approach were published as early as 1997 by Hosseini et al. (1997). Some variations include Kumar et al. (2021) who pair their NN with other classical ML approaches and Lui et al. (2023) who uniquely consider different *S. typhimirium* strains in a multitask model. This basic structure is also shared by many non-Ames NN-based quantitative structure activity relationships (QSARs), such as those for drug-induced liver injury (DILI) by Xu, Dai, et al. (2015) and Paykan Heyrati et al. (2023) and drug transport proteins like bile salt export pump (BSEP) (Kong et al. 2023). In our results section, we include a comprehensive overview of the performance of recent Ames prediction models reported in the 2nd Global Ames Prediction Challenge (Furuhama et al. 2023).

However, there are a number of problems with these existing MFs-based approaches.

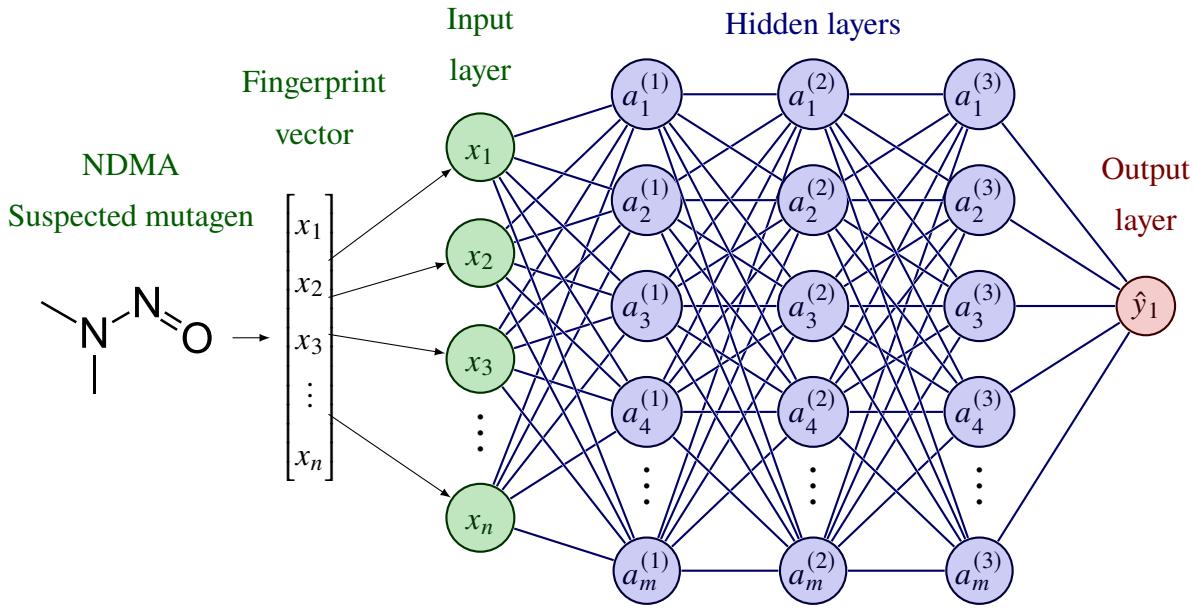


Figure 1.4: An archetypal neural network for Ames mutagenicity prediction. A molecule is transformed into a MF bit vector representation, which is fed as input to a neural network. The network's output layer is a binary prediction of the molecule's activity (i.e., Ames-positive). This basic structure is generalisable to many cheminformatics tasks.

Fingerprints struggle to capture rich structural information. MF usually describe local atomic substructures, not gestalt molecular shape. Substructural representations are often sufficient for predictions based on interactions within a binding pocket, but are less robust when considering binding site accessibility which requires representation of the whole-molecule electrostatic interactions (Hartenfeller and Schneider 2011). Thus, modelling whole-molecule interactions is challenging with MFs and often requires docking or other methodologies (Hartenfeller and Schneider 2011; Kearnes et al. 2016).

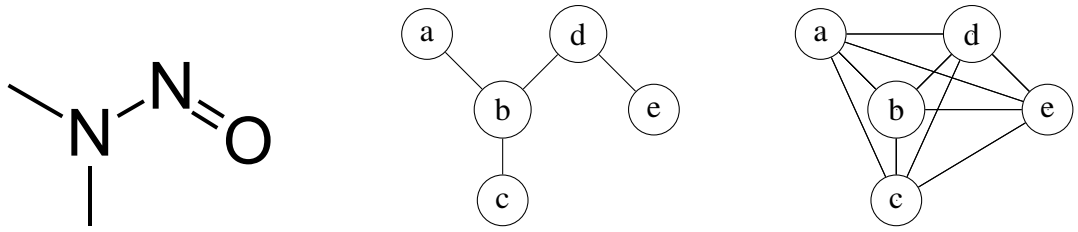
MF do not exhaustively describe molecular features. Many MFs were originally designed for classical ML techniques, such as logistic regression and support vector machines. These methods suffer from the curse of dimensionality, where computational costs increase exponentially with MF size (Keogh and Mueen 2010). Consequently, large MF bit vectors become computationally intractable and hinder model robustness (Altman and Krzywinski 2018). To mitigate these issues, MFs typically describe a limited set of features, such as local structures and atomic presences, to avoid high-dimensional or sparse bit vectors that degrade classical ML performance (Altman and Krzywinski 2018). As a result, MFs do not comprehensively capture every feature of every atom (Durant et al. 2002; Rogers and Hahn 2010). However, newer NN-based models largely mitigate the curse of dimensionality and maintain good performance with high-dimensional inputs. Therefore, classical fingerprints may omit information that could be beneficial for these advanced approaches (Krizhevsky et al. 2017; Zhang et al. 2016).

Fingerprints rarely encode quantity. Whilst some solutions to this problem are emerging, like those developed by Zhong and Guan (2023), MFs generally only encode the binary presence of atoms or groups in their vectors, not the quantity of such groups.

Hence, MF-based approaches are sub-optimal for cheminformatics tasks as they do not exhaustively describe the molecules they encode, omitting structural and quantity information despite the ability of contemporary architectures to handle such rich data.

1.6 Graph Neural Networks for Cheminformatics

These problems with MF can be resolved using graph-structured data and graph neural networks (GNNs). A graph $G = (V, E)$ models entities as a V set of v nodes, and a E set of pairwise entity relationships e, known as edges, in non-Euclidean space (Scarselli et al. 2009). The features of any node i are contained in a real-valued d-dimensional vector $\vec{h}_i \in \mathbb{R}^d$. Graph structures are thus analogous to molecules when we consider atoms as nodes and bonds as edges as in Figures 1.5a and 1.5b.



(a) NDMA structure (b) Molecular bipartite graph (c) Molecular complete graph

Figure 1.5: Representations of N-Nitrosodomethylamine (NDMA) as **a.** A skeleton diagram. **b.** A molecular bipartite graph, the input to many GNNs. **c.** A molecular complete graph, the underlying graph structure assumed by transformer models due to their attention layers.

A GNN is a NN operating on atom-like graph-structured data which may learn to predict the characteristics of nodes (e.g., atomic orbital hybridisation), edges (e.g., bond length), or the whole graph (e.g., Ames positivity) (Jin et al. 2022). To do this, GNNs iteratively update their node features each layer L times per full run through the data or *epoch*, p eq. (1.16). To update its representation $\vec{h}_i$, a node i first receives messages from its neighbours N_i via a *message function* (1.16). A permutationally-invariant function (e.g. arithmetic mean) then *aggregates* the messages received from N_i with the node's own feature vector $\vec{h}_i$ (1.17). Finally, $\vec{h}_i$ is updated to $\vec{h}'_i$ according to an *update* function (1.18). After the final epoch, P, *graph pooling* flattens all node and edge feature vectors into a single graph vector $\vec{h}_G$ (1.19) (Duval et al. 2023). Head NN layers take $\vec{h}_G$ as input and learn to classify the molecule as Ames-positive or -negative via minimisation of a loss function as in conventional NN backpropagation.

$$m_{ij} = Message(\vec{h}_i, \vec{h}_j) \tag{1.16}$$

$$\vec{A}_i = Aggregate(\vec{h}_i, \{m_{ij}\}_{j \in N_i}) \tag{1.17}$$

$$\vec{h}'_i = Update(\vec{A}_i) \tag{1.18}$$

$$\vdots$$

Repeat for L layers

$$\vec{h}_G^P = Readout(\{h_i\}_{i \in V}) \tag{1.19}$$

GNNs are ideally suited for cheminformatic tasks as they contain powerful inductive biases which solve many of the problems of MF-based models. Inductive biases are the assumptions different model architectures make about their input data (Baxter 2011).

GNNs innately capture the structural information of a molecule due to their relational inductive biases. This is because the structure of the network **is** the structure of the molecule (Kearnes et al. 2016). As GNNs nodes only aggregate the values of connected nodes, an atom or node is thus only influenced by those to which it is bonded (Bishnoi et al. 2022). This recapitulates the real-life importance of bonds in determining intramolecular interactions and local atomic environments. Non-graph MF approaches cannot natively capture this nuance.

GNNs exhaustively describe molecular features. As opposed to MFs, which describe a molecule using a single bit vector, GNNs describe each node and edge independently with a bit vector. This means that the whole molecule is a composition of multiple bit vector-described atoms and bonds. As a result, GNNs may describe all possible features of each atom and bond, including hybridisation state and bond angles which are seldom described in MFs.

GNNs always contain quantity information. As each node and edge is described by it's own bit vector, the number of atoms and bonds corresponds directly to the number of number of bit vectors associated with the graph.

The permutational invariance of the aggregation function also means that differing orders of atomic connection do not affect the final pooled graph, or the network's final output $\hat{y}$ (Duval et al. 2023). Furthermore, GNN architectures operating within Euclidian space (e.g., GemNet) produce molecular representations invariant to input translation, rotation, and dilation (Gasteiger et al. 2021). Thus, GNNs better represent the importance of atomic bonds and the invariance of pharmacological systems to non-structural molecular permutation than other approaches that do not mirror molecular structures in their underlying structures.

For these reasons, GNNs are a superior method of molecular representation. Hung and Gini (2021) demonstrated this empirically for the Ames mutagenicity prediction task by achieving state of the art (SOTA) performance in the Furuhama et al. (2023) Global Ames Prediction Challenge using a GNN.

GNNs have also achieved near-SOTA performance in other domains, such as DILI and BSEP inhibition prediction (Wu, Qian, et al. 2023; AbdulHameed et al. 2023). However, the general superiority of GNN models is perhaps best demonstrated by their leading performance on benchmark datasets like Tox21, as shown by Xiong, Wang, et al. (2020) and Gao et al. (2023).

(a) GNN message passing

(b) Graph attention network (GAT) message passing

Figure 1.6: **a.** The conventional GNN message passing schema wherein all elements of $\vec{h}_j$ have equal impact upon the updated $\vec{h}'_i$. **b.** The graph attention schema wherein elements of $\vec{h}_j$ differentially impact the updated $\vec{h}'_i$ based on their *attention coefficients* a or "importance".

The GAT, shown in Figure 1.6b, is an extension of the GNN to attention by Veličković et al. (2017), inspired by the landmark work of Vaswani et al. (2017). Attention enables differential weighting of feature vectors so the most pertinent features of a node's neighbours have a greater impact on the updated $\vec{h}'_i$, as shown in Figure 1.6b (Veličković et al. 2017). However, Veličković et al. (2017) implements a rudimentary *additive* attention mechanism as opposed to the better-performing multi-head scaled dot-product attention within transformers, shown in eq. (1.15). An implementation of this rudimentary graph attention mechanism represents the current SOTA in GNN-based Ames prediction (Hung and Gini 2021). Notably, no GNN-based Ames models currently incorporate transformer multi-head scaled dot product attention, leaving significant performance on the table.

1.7 AmesFormer, Our Hypotheses and Aims

The aim of this thesis is to move beyond this paradigm dominated by MF and simple graph models by constructing AmesFormer, a graph transformer network (GTN) for Ames mutagenicity prediction. A GTN combines the graph structure of GNNs with the attention of transformers. To do this, we leverage cutting-edge research by Chengxuan et al. (2021) who were the first to construct GTN which showed superior performance to earlier GNNs. We also aim to compare the performance of our Ames GTN with that of other models presented in a recent survey of Ames QSARs by Furuhama et al. (2023). Furthermore, we intend to deploy our GTN model on our lab website (https://cptlab.au/) for use by regulatory, industrial, and academic stakeholders, especially the Australian Industrial Chemicals

Introduction Scheme (AICIS), who have invested in our model development program in the past (FMH Ignition Grant 2019).

We hypothesise that a GTN capitalising on our lab's Ames domain knowledge will achieve SOTA or near-SOTA performance for the computational prediction of Ames mutagenicity. As transformers generally require more data to reach performance-parity with other model architectures, we also hypothesise that the existing Ames datasets from the literature will be sufficient to train a transformer of near-SOTA performance and that larger datasets will produce better performing models (Al-hammuri et al. 2023).

Chapter 2

Methods

2.1 Datasets

We train four separate GTNs, one on each dataset and one based on the best-performing of three. We utilise 80% of each dataset for model training, and the remaining 20% for validation and hyperparameter optimisation. The final evaluation of the performance of each model was performed using the Honma hold-out test dataset (n = 1590). We ensured no molecules were duplicated across the training, validation or test sets via Python set-based analysis.

The Hansen dataset (n = 6512) shows good class balance but is limited by its small size. The Honma set (n = 12 134) is larger and more regularly employed in recent Ames QSAR literature, but shows severe class imbalance, with only 14.4% of the contained compounds being Ames-positive (Furuhama et al. 2023; Hansen et al. 2009; Li, Liu, et al. 2023). We were kindly granted access to the proprietary Honma dataset by the National Institute of Health Sciences, Japan (NIHS-J) for the duration of this project. The combined dataset containing both the Honma and Hansen datasets is the largest (n = 20 241), and shows a reduced class imbalance of 28.2%. When building the combined dataset, we ensured there were no duplicate or contradictory entries.

Each dataset described compounds as Simplified Molecular-input Line-entry System (SMILES) strings. We transformed these SMILES to graphs using Pytorch-Geometric, featurising the nodes and edges of our molecular graphs with the RDKit functions described in Table 2.2 (Landrum 2012). We excluded explicit representation of hydrogen atoms from our graphs to save on computational resources in line with many previous GNN architectures (Xiong, Wang, et al. 2020; Jin et al. 2022).

Table 2.1: Datasets to be examined within this project.

Name	**Honma Dataset**	**Hansen Dataset**	**Combined Dataset**
Publication	Furuhama et al. (2023)	Hansen et al. (2009)	This thesis
Data source	NIHS-J	Six existing datasets	NIHS-J, Hansen et al. (2009)
Training size	12,134	6,512	20,241
Validation size[*]	1213	651	2024
Testing size[*]	1590	1590	1590
Percent Positive[**]	14.4	53.8	28.2
Construction	Duplicates, salts removed	Duplicates removed	Salts removed
Our Cleaning	Invalid SMILES removed	Not required	Not required
Use in Thesis	Modelling, testing	Modelling	Modelling

[*]Number of unique SMILES strings after data cleaning described in the *our cleaning* row.

[**]We consider class A and B compounds to be Ames-positive in accordance with Furuhama et al. (2023).

Table 2.2: RDKit atom and bond encodings used to featurise molecular graph nodes and edges.

Encoding	**Encoding Name**
Atom Encoding	
Atomic Number	GetAtomicNum()
Chirality	GetChiralTag()
Degree	GetTotalDegree()
Formal Charge	GetFormalCharge()
Hydrogen count	GetTotalNumHs()
Radical electron count	GetNumRadicalElectrons()
Hybridisation State	GetHybridization()
Aromatic bond (bool)	GetIsAromatic()
Part of a ring? (bool)	IsInRing()
Bond Encoding	
Bond type	GetBondType()
Stereoisomerism	GetStereo()
Conjugation	GetIsConjugated()

2.2 Exploratory Data Analysis

We perform exploratory data analysis on the Hansen, Honma and Combined datasets using 1028-bit Morgan fingerprints of radius two for each SMILES structure as suggested by Hernández and Ballester (2023).

We then performed 3-dimensional principal components analysis (PCA) to derive the three principal components of the fingerprints for visualisation purposes and report the variance explained by each principal component.

To cluster our molecules, we perform a 2-dimensional Uniform Manifold Approximation and Projection (UMAP) to group the MFs into seven clusters with Scikit-learn (1.4.2) and UMAP-learn (0.5.6). Morgan fingerprints (1024-bit, radius 2) were calculated from the SMILES strings for each molecule in the dataset (**hernandez-hernandez_best_2023**). UMAP, with parameters set to n_components=2, n_neighbors=100, and min_dist=0.0, was applied to these MFs. We used Jaccard distance as the metric to preserve the local structure of the data during dimensionality reduction (Hernández and Ballester 2023).

The Jaccard Index which measures the similarity of finite sets is defined as:

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

To assess the degree of inter-cluster chemical similarity, we calculated the Tanimoto distances between the original high-dimensional fingerprints **before** their projection onto the 2D UMAP space. This is necessary because the UMAP space is a Riemannian manifold and does not meaningfully define Euclidean or Jaccard distances (McInnes et al. 2018). As the distances in UMAP space optimised to preserve local neighborhood relationships, they distort global distances and make such traditional distance metrics inappropriate (McInnes et al. 2018). Our approach of calculating Tanimoto distance prior to UMAP projection avoids this problem, allowing us to meaningfully report inter-cluster distances.

The Tanimoto distance between two MFs A, B, introduced by Rogers and Tanimoto (1960), is defined as:

$$T(A, B) = \frac{|A \cap B|}{|A| + |B| - |A \cap B|} \tag{2.2}$$

$$\text{Tanimoto Distance} = 1 - T(A, B) \tag{2.3}$$

We then perform agglomerative clustering with Ward linkage on the 2-dimensional UMAP-transformed data to form seven clusters. The Ward linkage method minimises the total within-cluster variance, defined as the sum of squared differences from the cluster mean, leading to compact, circular

clusters (Ward 1963). Formally, it iteratively merges clusters A and B to minimise the increase in the total within-cluster variance:

$$\Delta E = \frac{|A| \cdot |B|}{|A| + |B|} \|\mu_A - \mu_B\|^2 \tag{2.4}$$

This method is suitable for clustering UMAP-reduced data as it effectively captures local structures and ensures cohesive clusters. Empirically, it also outperforms Taylor-Butina clustering, another popular method (Hernández and Ballester 2023). To quantify the quality of our clustering, we report the silhouette score which is defined as:

$$s = \frac{b - a}{\max(a, b)} \tag{2.5}$$

Where a is the mean intra-cluster Jaccard distance eq. (2.1) and b is the distance to the nearest cluster.

To visualise these clusters and provide insight into the chemical space of our datasets we present a matrix showing the mean pre-UMAP Tanimoto distances between the molecules of each UMAP cluster.

2.3 Architecture

We employ the Graphormer architecture (Figure 2.1) introduced by Chengxuan et al. (2021) to construct our GTN, AmesFormer. However, due to numerous deprecations and advancements in the Python libraries available for building GTN models, we elected to fully reimplement Graphormer using PyTorch (2.2.2) and PyTorch-Geometric (2.5.2) (Paszke et al. 2019; Fey and Lenssen 2019).

2.3.1 Attention

We implement the classic scaled dot-product multi-head attention seen in eq. (1.15). However, we place the layer norm before the attention calculation in line with more recent transformer architectures (Xiong, Yang, et al. 2020).

2.3.2 Structural encodings

AmesFormer implements three encodings specifically beneficial to cheminformatics tasks. These encodings are implemented as attention biases or priors.

Spatial encodings. As previously discussed, self-attention means transformers assume their input data is fully connected and unordered, as each token or atom attends to itself and all others (Vaswani

et al. 2017; Devlin et al. 2018). This poses a significant challenge in NLP, where the sequential order of words in a sentence is crucial for understanding its semantic implications. Hence, Vaswani et al. (2017) developed *positional encodings*. Positional encodings, added before the attention module, describe the absolute position of each token in the input, thus enabling the transformer to incorporate the positional relationships between tokens into its representations (Vaswani et al. 2017). Positional encodings empirically improve transformer performance, as without them, positional data is wholly unavailable to the model (Devlin et al. 2018).

AmesFormer features spatial encodings, a generalisation of positional encodings to graphs, which captures the spatial relationships between nodes. This resolves the graph transformer's deleterious assumption of fully connectedness and improves AmesFormer performance.

Formally, the *spatial encoding* is a function $\phi(v_i, v_j) \rightarrow \mathbb{R}$ representing the shortest path distance (SPD) between any two nodes, v_i, v_j in a graph G. The SPD is multiplied by a learnable scalar b (i.e., a scalar optimisable during back propagation), and added to the self-attention calculation as a bias term (2.6). The same biasing scalar b is shared between all layers.

$$A_{ij} = \underbrace{\frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}}}_{Self-attention} + \underbrace{b_\phi(v_i, v_j)}_{Spatial\ encoding} \tag{2.6}$$

As Chengxuan et al. (2021) does not detail the SPD algorithm they utilise, we implement the Floyd-Warshall algorithm as shown in algorithm 1. This choice was motivated by the ease of implementation and good time complexity $\Theta(|V|^3)$ on fully connected graphs (Floyd 1962). However, any algorithm to find the SPD on directed weighted graphs (e.g., Djikstra's algorithm) would suffice (Dijkstra 1959).

Centrality encodings. AmesFormer appends the in- and out-degree of each node to their feature vectors $\vec{h}_i$ (Chengxuan et al. 2021). This is termed the *centrality encoding*, where z^- and z^+ are learnable vectors related to the number of edges impinging upon, $\deg^+(v_i)$, and exiting from, $\deg^-(v_i)$, node v_i respectively (2.7) (Chengxuan et al. 2021). As our molecular graphs are undirected and atomic bonds have no inherent direction, our spatial encoding, in practice, encodes the *total* degree of each node (2.8).

$$\vec{h}_i = \vec{h}_i + z^-_{\deg^-(v_i)} + z^+_{\deg^+(v_i)} \tag{2.7}$$

$$\vec{h}_i = \vec{h}_i + z_{\deg(v_i)} \tag{2.8}$$

Centrality encodings further enrich the structural information AmesFormer may glean from input molecules.

Edge encodings. For cheminformatics tasks, edges carry valuable information, such as the types of bonds between two atoms. The edge encoding aggregates edge information along shortest path, $SP_{ij} = (e_1, e_2, \ldots, e_N)$, between each node pair (v_i, v_j) (Chengxuan et al. 2021). Specifically, it is a mean of the dot products between each edge feature along the path $\vec{e}_n$ and a corresponding learnable embedding w_n^E. The transpose $(w_n^E)^T$ facilitates the computation of the dot product with $\vec{e}_n$, the edge feature vector. The edge encoding effectively yields a bias term for the attention mechanism that embodies the collective edge features along the shortest path between all node pairs in the graph. We may think of edge encodings as allowing AmesFormer to better understand its local bond environment. With edge encodings, the final form of the AmesFormer attention calculation is:

$$A_{ij} = \underbrace{\frac{(h_i W_Q)(h_j W_K)^T}{\sqrt{d}}}_{Self-attention} + \underbrace{b_\phi(v_i, v_j)}_{Spatial\ encoding} + \underbrace{\frac{1}{N}\sum_{n=1}^{N} \vec{e}_n \cdot (w_n^E)^T}_{Edge\ encoding} \tag{2.9}$$

Ablation studies in Chengxuan et al. (2021) empirically show that these encodings enhance the performance of GTNs.

Algorithm 1 Floyd-Warshall Algorithm

```
1:  procedure FloydWarshall(W)                          ▷ W is the graph's adjacency matrix
2:      n ← the number of vertices in the graph
3:      let dist be a n × n array of minimum distances initialized to ∞
4:      for i ← 1 to n do
5:          dist[i][i] ← 0
6:      end for
7:      for each edge (u, v) do
8:          dist[u][v] ← w(u, v)                        ▷ w(u, v) is the weight of the edge (u, v)
9:      end for
10:     for k ← 1 to n do
11:         for i ← 1 to n do
12:             for j ← 1 to n do
13:                 if dist[i][k] + dist[k][j] < dist[i][j] then
14:                     dist[i][j] ← dist[i][k] + dist[k][j]
15:                 end if
16:             end for
17:         end for
18:     end for
19:     return dist
20: end procedure
```

2.3.3 Other elements of AmesFormer

Inspired by recent research, we make some further modifications to AmesFormer compared to Graphormer to improve computational efficiency and model performance.

We modify the Gaussian error linear unit (GELU) activation function. The original Graphormer employs the classical GELU activation function seen in eq. (2.10) where $\Phi(x)$ represents the cumulative distribution function for a Gaussian distribution and $\text{erf}(\cdot)$ is the Gauss error function (Hendrycks and Gimpel 2016). However, this formulation is computationally expensive as $\text{erf}(\cdot)$ requires calculation of a non-elementary integral of the Gaussian function. Thus, we instead use the approximate GELU function shown in eq. (2.11); sacrificing some exactness to improve the feed-forward model training speed (Hendrycks and Gimpel 2016).

$$\text{GELU}(x) = x \cdot \Phi(x) = x \cdot \frac{1}{2}\left[1 + \text{erf}\left(\frac{x}{\sqrt{2}}\right)\right] \tag{2.10}$$

$$\text{Aprox\_GELU}(x) = 0.5x\left(1 + \tanh\left(\sqrt{\frac{2}{\pi}}\left(x + 0.044715x^3\right)\right)\right) \tag{2.11}$$

We implement the GreedyLR learning rate scheduler. Graphormer originally implements a linear LR scheduler, as shown in algorithm 3. This scheduler linearly interpolates the learning rate between a given start value, η_{start} and end value, η_{end}. Hence, the LR $\eta(t)$ at epoch t is described with the formula:

$$LR(t) = \eta(t) = \eta_{\text{start}} + \left(\frac{t}{N}\right) \cdot (\eta_{\text{end}} - \eta_{\text{start}}) \tag{2.12}$$

For AmesFormer, we instead utilise GreedyLR by Subramanian and Ganapathiraman (2023), a modern learning rate scheduler which adapts to the gradient of the loss surface. GreedyLR scales the LR based on short-term trends in the loss. It also incorporates a *patience* parameter to delay changes in the LR, making it robust to unexpectedly rough locales of the loss surface (Subramanian and Ganapathiraman 2023). GreedyLR demonstrates superior performance on various benchmark tasks (Subramanian and Ganapathiraman 2023). See algorithm 4 for further details.

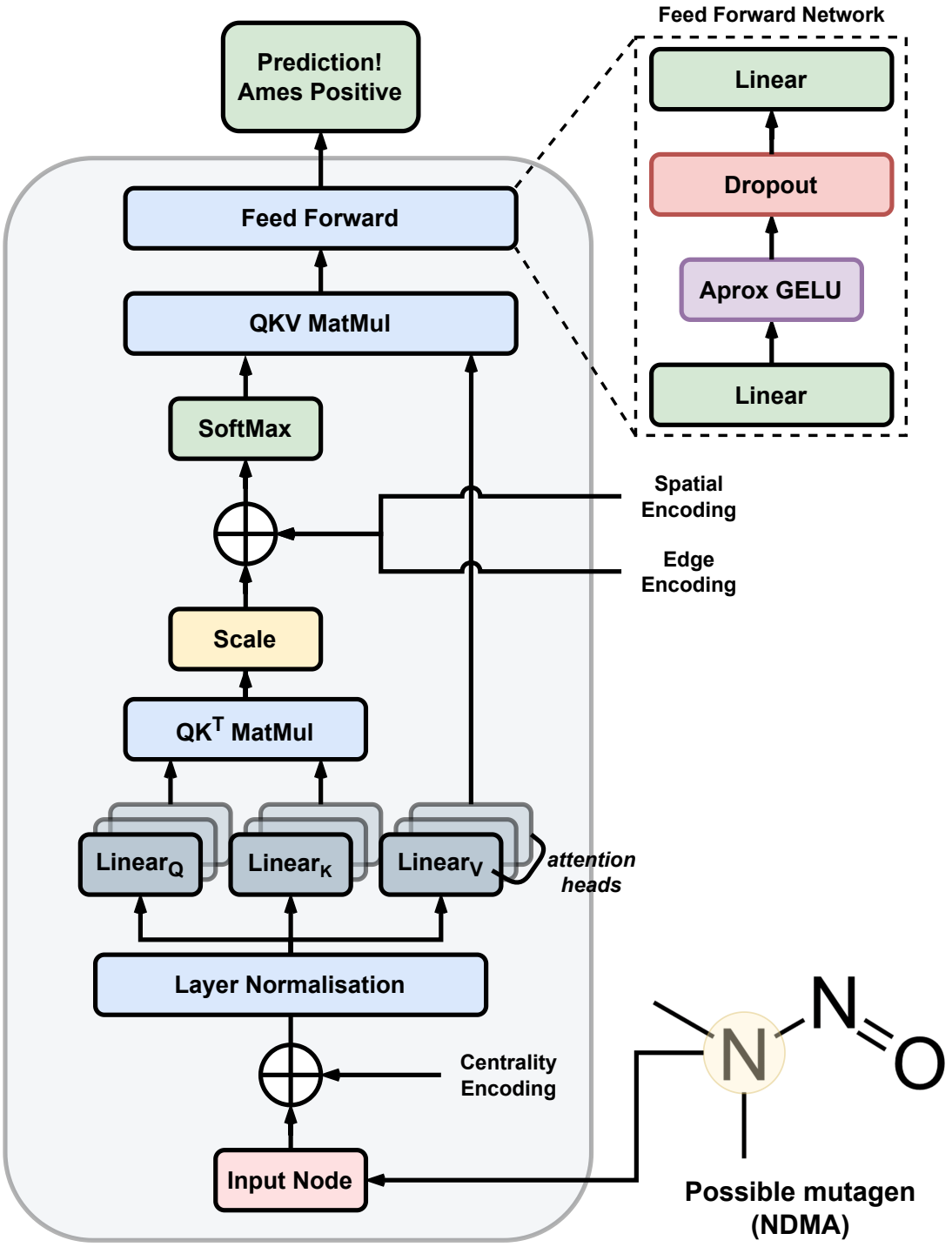


Figure 2.1: An illustration of the complete AmesFormer architecture. AmesFormer takes the node features of a given potential mutagen as input, combining them with the centrality encoding (eq. (2.8)). Layer normalisation is then applied before the multi-head scaled dot product attention module (eq. (1.15)). The raw attention is calculated using QK^T (eq. (1.14)), and the spatial encoding (eq. (2.6)) and edge encoding (eq. (2.9)) are added as biases. After applying softmax, the result is multiplied with the values matrix QKV. The attention output is then passed through a feed-forward neural network using the approximate GELU function with dropout. Finally, after seeing every node of the graph, a linear layer classifies (predicts) the molecules as Ames-positive or negative.

2.4 Training

One model was trained per dataset. We name them AmesFormer-Hansen, AmesFormer-Honma and AmesFormer-Duo. The best-performing of these three models was chosen for retraining with both the original training and validation sets to maximise available data and hence its performance on the test set. We save a model snapshot or ‘‘checkpoint’’ each epoch, and save the snapshot which produced the lowest validation loss. We term this the *best loss* model.

We train each AmesFormer with three transformer blocks. Each block contains a feed forward neural network (FFN) with a hidden dimension d of 80 and dropout of 0.1. The FFNs use the approximate GELU activation function described in eq. (2.11). The multi-headed scaled dot product attention modules contain four heads, each with a hidden hidden dimension of 128 and an attention dropout of 0.1. The weight matrices of the FFNs and attention modules are randomly initialised using He initialisation sampled from a normal distribution as seen in eq. (1.2). These parameters resulted in a final model size of 264,696 parameters.

We train each AmesFormer using binary cross entropy loss (BCE Loss) for 120 epochs with a batch size of 32 and no gradient accumulation steps on an NVIDIA Titan V graphics processing unit (GPU) supplied in an NVIDIA Accelerated Data Science Grant.

To arrive at these hyperparameters we used Optuna (3.6.1) to perform Bayesian hyperparameter optimisation (Akiba et al. 2019). 300 trials with hyperband pruning were run, allowing early termination of poorly-performing hyperparameter sets, thus reducing total optimisation time (Li, Jamieson, et al. 2016).

We use the AdamW optimiser, a weight-regularised version of Adam by Kingma and Ba (2014), which guards against over-fitting (Loshchilov and Hutter 2017). The AdamW ϵ was set to 1×10^{-8} and $(\beta 1, \beta 2)$ to (0.9, 0.999). Gradient clipping, which limits gradient sizes during training to prevent exploding gradients, was set to 5.0. See Table 2.3 for an exhaustive overview of AmesFormer hyperparameters.

We set the minimum and maximum LRs of the AmesFormer GreedyLR scheduler to 1×10^{-6} and 1×10^{-3}, respectively. We enable the between LR adjustments and set a patience parameter of 4.0. A complete overview of GreedyLR parameters is available in Table 2.4.

Table 2.3: Comparison of hyperparameters for Graphormer and AmesFormer(Chengxuan et al. 2021).

Hyperparameters	**Graphormer (Chengxuan et al. 2021)**	**AmesFormer***
Layers	12	3
FFN Hidden Dimension d	768	80
FFN Dropout	0.1	0.1
Attention Heads	32	4
Head Hidden Dimension	24	128
Attention Dropout	0.1	0.1
Max Epochs	300	120
Activation Function	GELU (2.10)	Aprox_GELU (2.11)
Readout Function	[Vnode]	[Vnode]
Loss function	L1 Loss	BCE Loss
Peak Learning Rate	2e-4	1e-3
Minimum Learning Rate	N/A	1e-6
Batch Size	1024	32
Learning Rate Scheduler	Linear	GreedyLR
Adam ϵ	1e-8	1e-8
Adam (β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)
Gradient Clip Norm	5.0	5.0
Weight Decay	0.0	0.0

*Includes all four AmesFormer models.

Table 2.4: The parameters for the GreedyLR learning rate scheduler used to train AmesFormer.

GreedyLR Parameter	**Value**
Patience	4.0
Cooldown	2.0
Minimum	1×10^{-6}
Maximum	1×10^{-3}
Warmup	2.0
Smoothing factor	True
Window	10.0
Scaling factor	0.5
Reset at epoch	Unused

All hyperparameters and GreedyLR parameters were read from a .toml file, modifications can also be made when calling the training script using flags implemented with the Click Python package (8.1.7).

2.5 Statistical Methodologies

All of the following performance measurements were made on the best loss models, which are the model snapshots which produced the lowest validation loss. This is essential as machine learning models often start to overfit their training data during extended training sessions, meaning the final version may not perform optimally on real-world tasks (Guo et al. 2017). The results reported are always performance on the held-out Honma test set, in line with Furuhama et al. (2023).

2.5.1 Performance Measurement

We assess the performance of AmesFormer using balanced accuracy and F1 score:

$$\text{Balanced Accuracy} = \frac{1}{2}\left(\frac{TP}{TP+FN} + \frac{TN}{TN+FP}\right) \tag{2.13}$$

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{2.14}$$

Where:

- TP (True Positives): Number of positive cases correctly identified as positive.
- TN (True Negatives): Number of negative cases correctly identified as negative.
- FP (False Positives): Number of negative cases incorrectly identified as positive.
- FN (False Negatives): Number of positive cases incorrectly identified as negative.
- Precision: The ratio of correctly predicted positives to the total predicted positives, defined as $\frac{TP}{TP+FP}$.
- Recall: The ratio of correctly predicted positives to all observations in the actual class, defined as $\frac{TP}{TP+FN}$.

Balanced accuracy (BA), shown in eq. (2.13), is the percentage of molecules that the model correctly predicts, adjusted to ensure equal weighting of the model's performance on both the majority and minority classes (Brodersen et al. 2010). As our dataset shows class imbalance, accuracy alone would

be a misleading performance metric (Provost et al. 2001). The F1 score eq. (2.14) is the harmonic mean of recall and precision (Hicks et al. 2022). Similarly to BA, F1 is robust to unbalanced datasets such as the Honma dataset (Hicks et al. 2022). It is also suited to regulatory toxicological tasks such as Ames mutagenicity where both false positives and false negatives have serious implications (Liu, Patlewicz, et al. 2017; Hicks et al. 2022). This balance is crucial in toxicology, where the cost of misclassification can be high, either by failing to identify a toxic compound (false negative) or mistakenly labeling a safe compound as toxic (false positive).

2.5.2 Calibration Analysis

In the following calibration analysis, we set the model output to logits, representing the probability the model ascribes to a given molecule being Ames-positive. Calibration reflects how accurately a model's predicted probabilities reflect the true likelihood of an outcome. For example, if a well calibrated model outputs an 80% probability of an event occurring, the event should actually be observed 80% of the time over many predictions.

We assess the calibration of AmesFormer using a calibration curve and expected calibration error (ECE), which measures the difference between the model's probability prediction and the actual outcome. ECE may be denoted as:

$$\mathrm{ECE} = \sum_{i=1}^{N} \frac{|B_i|}{n} \left|\mathrm{acc}(B_i) - \mathrm{conf}(B_i)\right| \tag{2.15}$$

To compute the ECE, we divide the predicted probabilities into N equal bins. For each bin B_i, we calculate the accuracy, which is the proportion of correct predictions, and the confidence, which is the average predicted probability. The ECE is the weighted average of the differences between the accuracy and confidence for each bin. The weights are the proportions of samples in each bin relative to the total number of samples. This means that when ECE is low the model is well calibrated because its predicted probabilities more closely align with the true likelihood of events.

We elected not to use the more common Brier score to assess calibration as it is only comparable across models when the refinement loss, one of three components of the Brier score, is the kept the same between each model (Flach and Matsubara 2007). This constraint is impractical to enforce with our analysis library, Scikit-learn 1.4.2.

2.5.3 Uncertainty Estimation

We perform Monte Carlo dropout to generate probabilistic predictions and sample our model's uncertainty (Gal and Ghahramani 2016). Monte Carlo dropout involves randomly setting the parameters of

a percentage of neurons to zero, resulting in a different prediction each sample. This can be viewed as a Gaussian approximation of a Bayesian process, where the dropout mask $\mathbf{M}$ (whether a neuron is zeroed or not) follows a Bernoulli distribution with p being the probability that a neuron is kept active (Gal and Ghahramani 2016). This may be denoted as:

$$P(\mathbf{M}) = \begin{cases} 1 - p & \text{for } \mathbf{M} = 0, \\ p & \text{for } \mathbf{M} = 1, \end{cases} \tag{2.16}$$

Thus, for each of our 1000 samples of the held-out Honma test dataset, a different 10%, p, of the neurons are zeroed yielding a distribution over the model's predictions that reflects its uncertainty.

Monte Carlo dropout helps gauge the performance of AmesFormer in the regulatory context as it displays the uncertainty of the model when sampling out-of-distribution new chemicals, such as those it would encounter in practice. The uncertainty measure also allows for more robust assessment of chemical safety by taking into account the *confidence* of the model in its predictions. Consider a molecule yielding an Ames-negative result but accompanied by high uncertainty. Such a compound warrants additional investigation, as it likely represents an out-of-distribution molecule for which the training data is sparse, thereby rendering its true Ames-positivity indeterminate.

We use Monte Carlo dropout to produce 95% confidence intervals (CIs) for the BA, F1 scores and ECE scores, to construct a BA and F1 box plot, and to plot confidence intervals on the calibration curve (Niculescu-Mizil and Caruana 2005). Such uncertainty quantification in QSAR models is explicitly recommended by the Organisation for Economic Co-operation and Development (OECD) (OECD 2014).

2.5.4 Selection of the Best AmesFormer

We perform 1 000 Monte Carlo dropout samples on AmesFormer-Hansen, AmesFormer-Honma and AmesFormer-Duo, producing 1 000 BA and 1 000 F1 scores for each AmesFormer model. To select the best performing of these models, we perform a non-parametric Conover-Friedman test followed by a post-hoc Holm-stepdown test with Bonferroni adjustments on both sets of 1 000 measurements (Conover and Iman 1979; Demsar 2006). We measure performance on the held-out Honma test set (Furuhama et al. 2023).

The Friedman test ranks the models across multiple datasets and whether any observed differences are statistically significant (Conover and Iman 1979). The Holm-stepdown test, applied post-hoc, controls the family-wise error rate by sequentially adjusting the p-values using the Bonferonni-Holm method (Holm 1979). Whilst the standard Bonferroni correction method adjusts each p-value uniformly,

the Bonferroni-Holm method compares each ordered p-value to an incrementally adjusted threshold, starting with the smallest p-value (Holm 1979). In most cases, the Holm-stepdown method is more powerful than Sidak or Bonferonni-adjustments alone, and is recommended for use in ML studies (Demsar 2006; Abdi 2010).

A parametric multiple-pairwise analysis of variance (ANOVA) and post-hoc test is unsuitable for our purposes as machine learning performance metrics frequently violate assumptions of normality, homogeneity of variance and independence of observations(Demsar 2006).

2.5.5 Performance Comparison With the Literature

After determining the best-performing model among AmesFormer-Hansen, AmesFormer-Honma and AmesFormer-Duo using the Friedman test we modify it in two ways to enable comparison with the literature. We first retrain the best-performing model without a validation set. This enables us to use all the available data for training, maximising performance. Secondly, when we generate the final BA and F1 scores for comparison with the Furuhama et al. (2023) data, we disable Monte Carlo dropout, ensuring our model's full performance is available for the task. We dub this model AmesFormer-Pro.

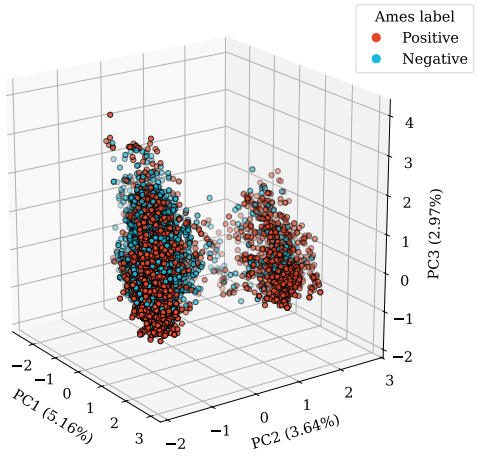
Chapter 3

Results

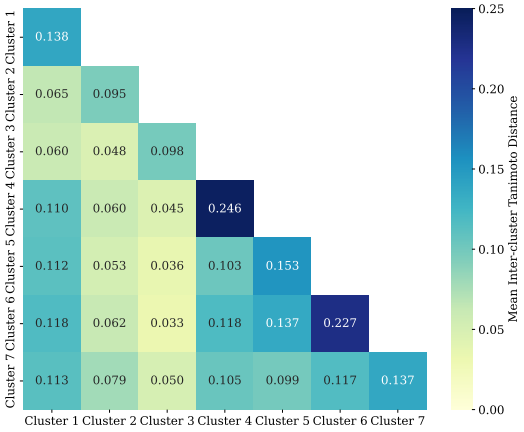
3.1 Exploratory Data Analysis

The first three principal components of the Hansen dataset explain 11.77% of variance in the Morgan fingerprints. The Hansen dataset also showed the clearest separation of Ames-negative and Ames-positive molecules, with a higher PC2 score correlated with Ames positivity. The PCA of the Honma and Combined datasets found their MF were not easily linearly separable, with the first three principal components explaining only 9.06% amd 9.23% of the variance, respectively.

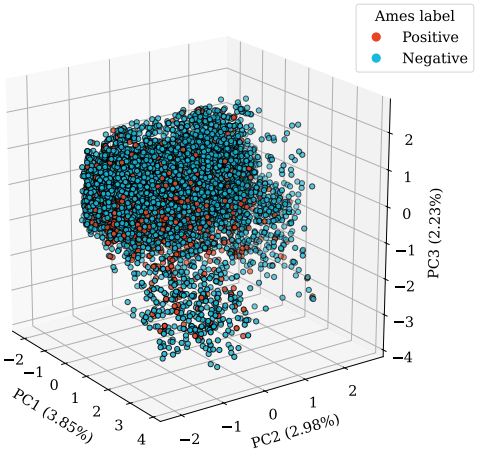
A higher silhouette score means that the data forms more distinct clusters (Rousseeuw 1987). The Hansen dataset showed the highest UMAP cluster silhouette score s of 0.488, indicating that the clusters formed are distinct and compact. Comparatively, the Honma and Combined datasets had showed s of 0.378 and 0.384, respectively. As seen in Figure 3.1, all datasets formed between two and three good clusters, as seen by the darker blue elements along the matrix diagonal which show high intra-cluster Tanimoto similarity. This suggests that each dataset is characterised by a few predominant chemical motifs around which the compounds are closely related.



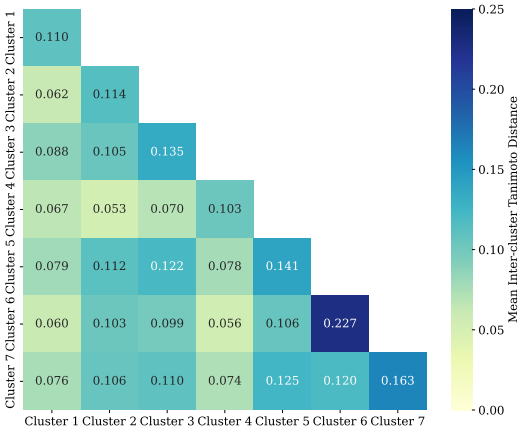
(a) Hansen dataset PCA.



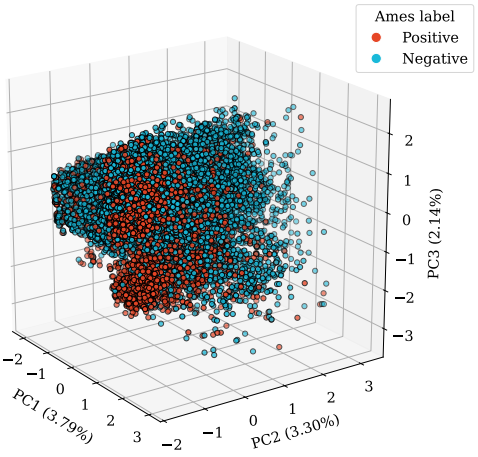
(b) UMAP clusters of the Hansen dataset



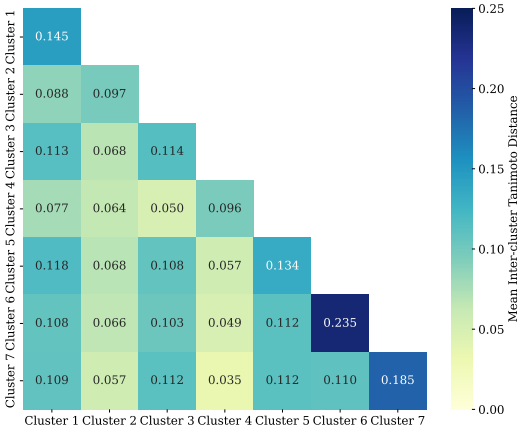
(c) Honma dataset PCA.



(d) UMAP clusters of the Honma dataset.



(e) Combined dataset PCA.

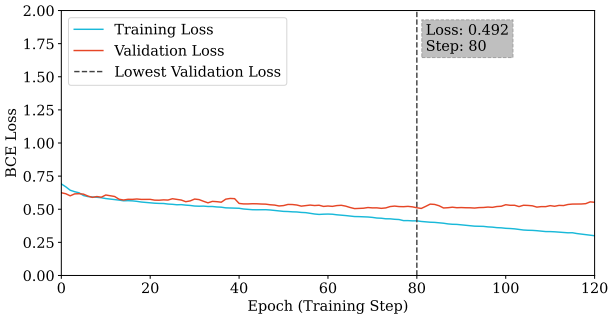


(f) UMAP clusters of the Combined dataset.

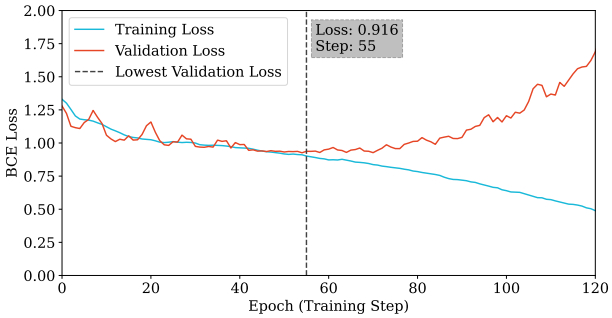
Figure 3.1: Left: 3-dimensional PCA of the molecules in each dataset represented as Morgan fingerprints (1028 bit, radius 2). The axes (PC1, PC2, PC3) represent the principal components, indicating the directions of maximum variance in the data, with the variance explained in parentheses. Ames-positive compounds are coloured red and Ames-negative compounds are coloured blue. Right: Matrices showing the mean pre-projection Tanimoto similarity between molecules of a UMAP cluster pair. Darker blue indicates greater similarity.

3.2 Training AmesFormer

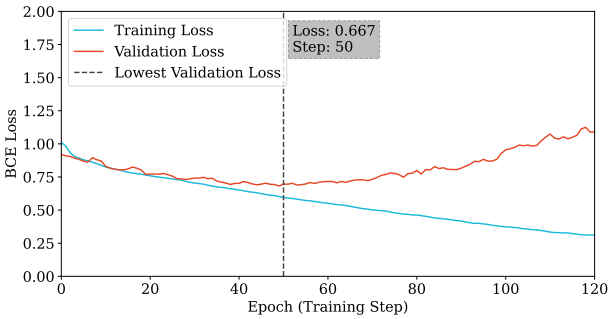
The training performance of AmesFormerwas evaluated using the training and test BCE Loss over 120 epochs. We apply rolling average smoothing (window = 3) to aid loss trend analysis. The grey dashed line marks the epoch which produced the model with the lowest validation loss and the point at which the training and validation loss trends diverged.



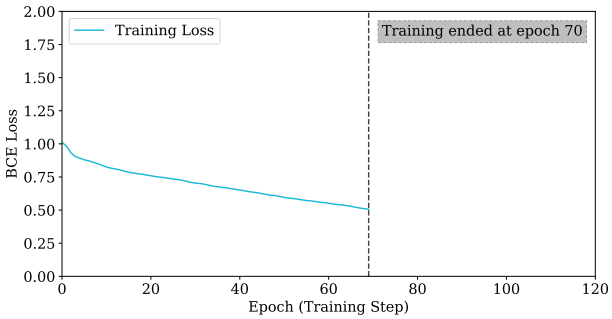
(a) Training and validation loss for AmesFormer-Hansen. The minimum loss of 0.492 was achieved at epoch 80.



(b) Training and validation loss for AmesFormer-Honma. The minimum loss of 0.916 was achieved at epoch 55.



(c) Training and validation loss for AmesFormer-Duo. The minimum loss of 0.667 was achieved at epoch 50.



(d) Training loss for AmesFormer-Pro. Training ended at 70 epochs and no validation loss was recorded.

Figure 3.2: Mean training loss (blue) and validation loss (orange) of each AmesFormer over 120 epochs, using BCE Loss loss with a rolling average smoothing window of three. The grey dashed line marks the epoch with the lowest validation loss, where a snapshot of the model was saved for further analysis and testing on the held-out Honma dataset. The inflection point indicates the onset of overfitting, where training loss decreases and validation loss increases.

Initially, the training loss declined gradually across all models; however, the epoch at which the training and validation losses began to diverge varied among the models (Figure 3.2). AmesFormer-Hansen showed the latest divergence at 80 epochs, achieving a minimum BCE Loss loss of 0.492 at this point (Figure 3.2a). The behavior of AmesFormer-Honma (Figure 3.2b) and AmesFormer-Duo (Figure 3.2c) was more similar. AmesFormer-Honma began to diverge at epoch 55, producing a best

loss of 0.916, while AmesFormer-Duo diverged at epoch 50, achieving a best loss of 0.667. These *best loss* models were saved and used to obtain the results presented later.

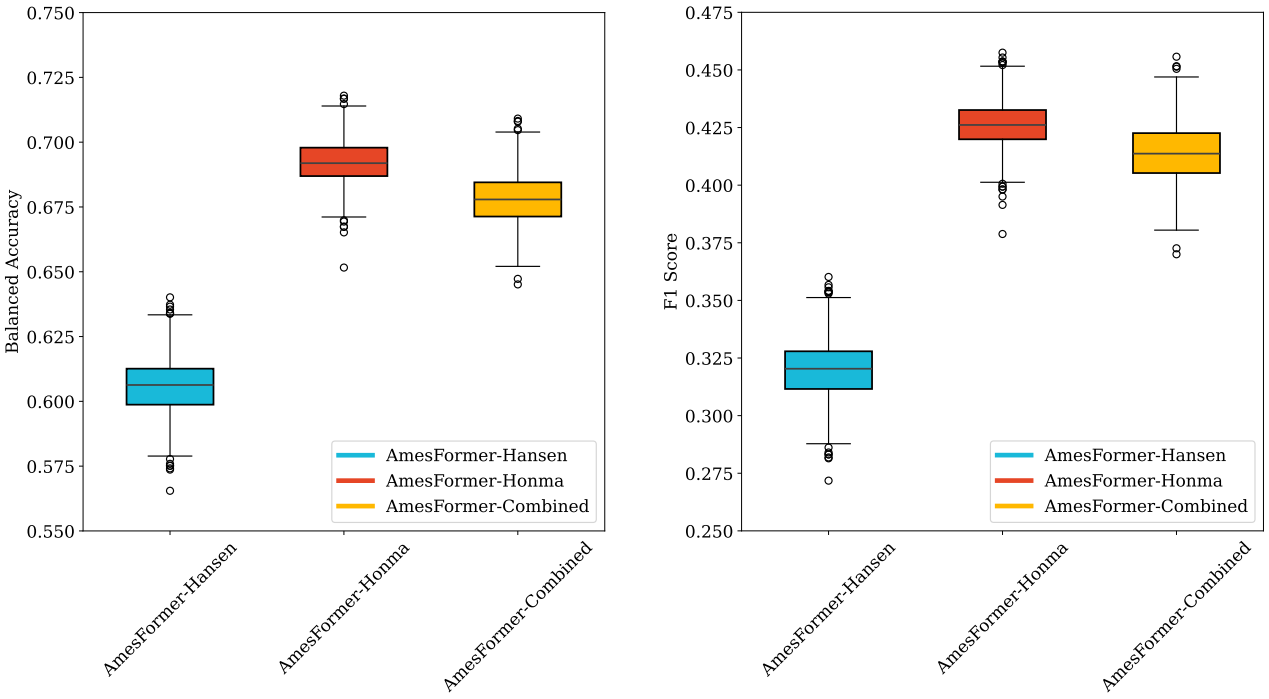
3.3 AmesFormer Performance

Amongst the three initial models, AmesFormer-Honma demonstrated the highest mean BA of 69.2% and an F1 score of 0.426 on the held-out Honma test set. In comparison, AmesFormer-Duo performed slightly worse, with a BA of 67.8% and an F1 score of 0.414. AmesFormer-Hansen was the weakest performer, yielding a BA of 60.6% and an F1 score of 0.320 on the held-out Honma test set.

Table 3.1: The mean performance of each best-loss AmesFormer over 1000 Monte Carlo dropout samples on the held-out Honma test set. We indicate the maximum error of estimation for the 95% confidence intervals ($1 - \alpha$).

Model	**AmesFormer-Hansen**	**AmesFormer-Honma**	**AmesFormer-Combined**	**AmesFormer-Pro**
Mean BA (%)	60.6 ±0.1	69.2 ±0.1	67.8 ±0.2	**74.0**
Mean F1	0.320 ±0.1	0.426 ±0.1	0.414 ±0.2	**0.479**
ECE	0.196 ±0.159	0.197 ±0.123	**0.157** ±0.154	0.200
Best epoch	80	55	50	N/A
Best validation loss	0.492	0.916	0.667	N/A

The test Friedman test revealed statistically significant differences between the performance of all three AmesFormer models on the Honma test set ($p < 0.0001$, $t = 1773.800$), providing strong evidence against the null hypothesis of there being no significant differences between the groups. A post-hoc Conover-Friedman test with Holm p-adjustments further confirmed highly significant differences between all model pairings ($p < 0.0001$).



(a) BA scores for the best loss AmesFormer models.

(b) F1 scores for the best loss AmesFormer models.

Figure 3.3: Boxplots of BA and F1 scores for each AmesFormer generated with 1000 Monte Carlo dropout samples. AmesFormer-Hansen is shown in blue, AmesFormer-Honma in red, and AmesFormer-Duo in yellow. The central line represents the median, while the box spans the interquartile range (IQR) from the second to the third quartile. Whiskers extend to the first and fourth quartiles, with outliers displayed as circles beyond the whiskers.

The superiority of AmesFormer-Honma in terms of BA can also be seen in Figure 3.3a. Note the slightly increased variance of AmesFormer-Duo in both Figure 3.3a and Figure 3.3b compared to AmesFormer-Honma.

Given the inter-group differences confirmed by the Friedman significance testing, and the superior performance of AmesFormer-Honma in terms of BA and F1 scores (Table 3.1), we selected AmesFormer-Honma as the basis for AmesFormer-Pro.

3.4 Performance Comparison With the Literature

We retrained AmesFormer-Honma to generate AmesFormer-Pro for comparison with the literature and evaluated its performance using BA and F1 score on the held-out Honma test set. AmesFormer-Pro ranked third in BA among the 21 models that reported results in the 2nd Global Ames Challenge, as shown in Table 3.2 (Furuhama et al. 2023). This represents a 3.9% improvement in BA over the

previous results from our lab, achieved by the DRSpicySTiM-Ensemble model (Furuhama et al. 2023). AmesFormer-Pro ranked eighth in terms of F1 score.

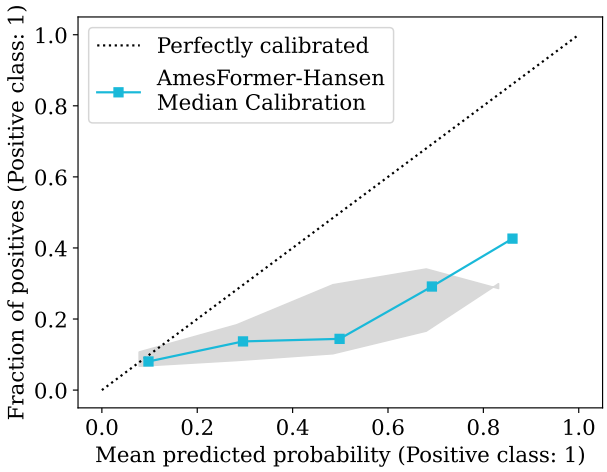
Table 3.2: A comparison of the performance of AmesFormer with models presented in Furuhama et al. (2023) on the held-out Honma test set. Our results for AmesFormer-Pro are shown in bold.

Team or Institution Name	**Model Name**	**BA (%)**	**F1 Score**
MN-AM	ChemTunes. ToxGPS Ames NIHS$_v$2	78.5	0.538
Meiji Pharmaceutical University	MMI-STK2	77.0	0.524
Our result	**AmesFormer-Pro**	**74.0**	**0.479**
Instem	Leadscope Consensus Model	73.7	0.497
LMC Bourgas University	TIMES_AMES 17.17.3	73.3	0.511
Altox Ltd.	GeneTox-iS	72.6	0.500
Evergreen AI, Inc.	Avalon	71.9	0.485
MultiCASE Inc.	PHARM_BMUT V1.8.0.0.17691.350	71.2	0.497
Simulations Plus Inc.	S+MUT_NIHS_ABC	71.2	0.421
The University of Sydney	DRSpicySTiM-Ensemble	70.1	0.425
Lhasa Ltd.	Sarah Nexus v.3.0.1 (2068 chemicals)	69.0	0.410
NCTR/FDA	DeepAmes	69.1	0.476
IRFMN	CONSENSUS (18k) V0.9.1	68.1	0.402
Liverpool John Moores University	DL	68.7	0.403
NIBIOHN	GNN(kMoL)_bestbalanced	67.2	0.470
SIOC, CAS	CISOC-PSMT (SIOC, CAS, China)	66.4	0.393
Politecnico di Milano	GCN	65.8	0.444
IdeaConsult Ltd.	AMBIT DeepN v4.85	65.6	0.408
Massachusetts Institute of Technology	Chemprop	64.3	0.420
Chemotargets	CHMT_GBoostSC	64.3	0.414
ISS	Mutagenicity ISS-modified2020	62.8	0.348
Gifu University	xenoBiotic 0.9q	60.3	0.334

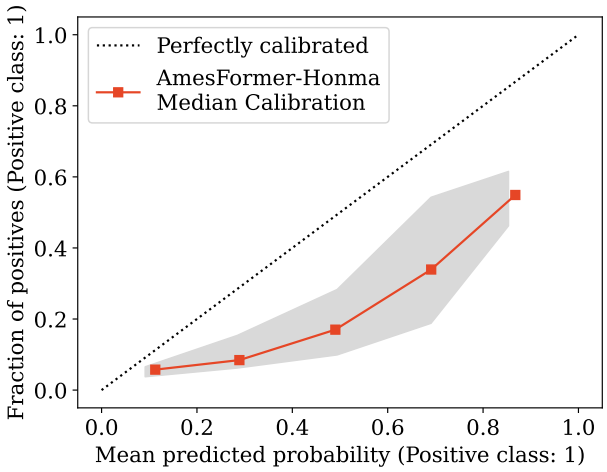
3.5 Uncertainty Estimation and Calibration

AmesFormer-Hansen exhibited an ECE of 0.196, with a 95% CI ranging from 0.076 to 0.394 (Figure 3.4a). This wide confidence interval indicates considerable variability in the calibration performance,

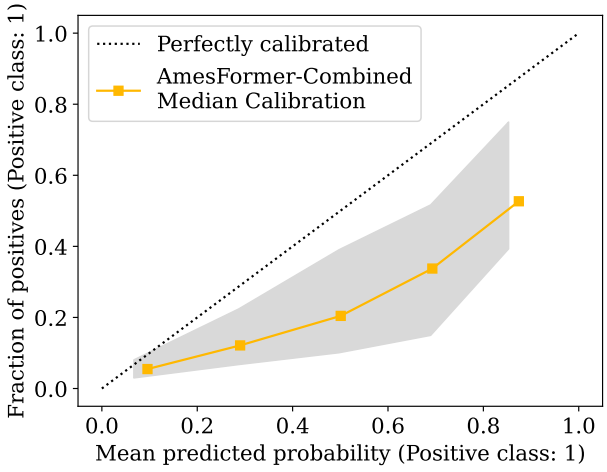
suggesting that the model's predicted probabilities are not consistently aligned with the true outcomes across the dropout runs. Similarly, AmesFormer-Honma produced an ECE of 0.197, with a 95% CIs of 0.087 and 0.333. Despite the similar ECE values between the Hansen and Honma models, the confidence intervals suggest AmesFormer-Honma is somewhat more reliable.



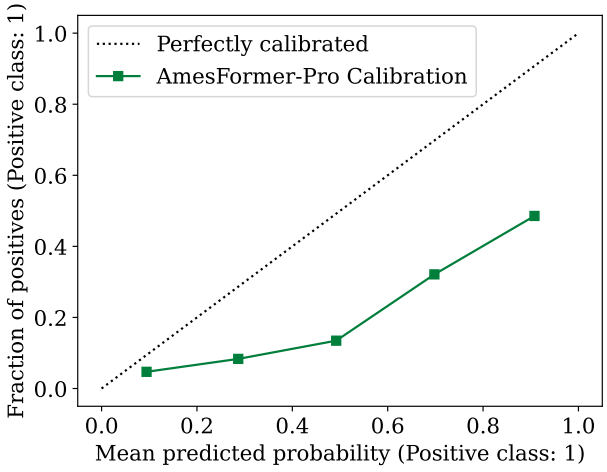
(a) The median calibration curve of AmesFormer-Hansen over 1000 Monte Carlo dropout samples with an associated ECE of 0.196 (95% CI: 0.076, 0.394).



(b) The median calibration curve of AmesFormer-Honma over 1000 Monte Carlo dropout samples with an associated ECE of 0.197 (95% CI: 0.087, 0.333).



(c) The median calibration curve of AmesFormer-Duo over 1000 Monte Carlo dropout samples with an associated ECE of 0.157 (95% CI: 0.044, 0.351).



(d) The median calibration curve of AmesFormer-Pro with an associated ECE of 0.200. No Monte Carlo dropout was performed.

Figure 3.4: Calibration curves for each AmesFormer on the held-out Honma test set. We plot the median calibration curve across 1000 Monte Carlo dropout samples with 95% CIs. The dashed line represents perfect calibration: a model that would output probabilities precisely corresponding to the actual observed frequencies of the events in the data. No CIs are shown for Figure 3.4d as Monte Carlo dropout was not performed.

AmesFormer-Duo, trained on both the Hansen and Honma datasets, yielded an ECE of 0.157, with a 95% confidence interval ranging from 0.044 to 0.351. Although the ECE for the combined dataset is lower than for the individual datasets, indicating improved average calibration, the wide confidence interval still points to considerable variability in calibration performance.

The worst performing model in terms of calibration was AmesFormer-Pro, with an ECE of 0.200. However, the CIs for the other models make an absolute statement of its classification inferiority challenging. CIs are not presented for this model as inference was performed without Monte Carlo dropout to maximise performance.

Chapter 4

Discussion

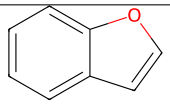
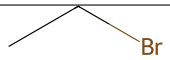
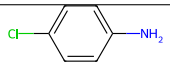
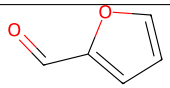
4.1 Understanding Our Datasets

Higher Ames-positivity means less chemical diversity. Chemical diversity refers to the range of different chemical structures within a dataset. As the Hansen dataset showed the best clustering (s = 0.488) and clearest principal components we can surmise that it had the least chemically diversity. This is because the compounds within each Hansen cluster are highly similar to each other and differ largely from those in the other six clusters. Thus, the overall diversity is limited as the compounds may be grouped into only seven relatively well-formed groups.

This may be because the Hansen dataset contains more Ames-positive molecules (53.8%) than the Honma (14.4%) and Combined (28.2%) datasets, and it thus contains more chemical substructures correlated with Ames mutagenicity. These substructures, such as those presented in Table 4.1, are then are then over-represented within Honma dataset, leading to molecules being easy to cluster based on the presence of these repeat Ames-correlated substructures (Tennant and Ashby 1991). This hypothesis is supported by the PCA data in Figure 3.1b, as the molecules showing a higher principal component two score were substantially more likely to be Ames-positive. This suggests that some chemical substructure(s) represented within PC2 are associated with Ames-positivity.

Notably, the presence of these substructures alone is not sufficient to predict mutagenicity, hence, we require models such as AmesFormer to detect more subtle structural differences.

Table 4.1: Table of some chemical substructures commonly associated with Ames-positive molecules (Tennant and Ashby 1991).

Molecule Name	Structure	SMILES
Benzofuran		C1=CC=C2C(=C1)C=CO2
Bromoethane		CCBr
4-Chloroaniline		C1=CC(=CC=C1N)Cl
Furfural		C1=COC(=C1)C=O

Because the PCA and UMAP are projections of the MFs to lower-dimensional spaces, we cannot simply count the number of repeat substructures within each dataset and consider the most common moities as likely principal components.

4.2 Insights From Training AmesFormer

Despite operating on the smallest dataset AmesFormer-Hansen took the longest time to begin divergence, and hence the longest to reach ‘‘convergence’’ (Figure 3.2a). According to the classical understanding of the bias-variance trade-off, optimal model performance is achieved when a model is expressive enough to capture and learn the trends in the data, yet not so complex that it overfits and memorises the training data (Rocks and Mehta 2022). Models that are excessively large and prone to memorisation are considered overparameterised. Overparameterised models not only tend to overfit the training data but also require more computational resources and longer training times to converge due to the increased complexity in optimising a larger number of parameters, especially when the amount of data is relatively small (Zhang et al. 2016). Since the same parameter count was used for all AmesFormer models, despite AmesFormer-Hansen being trained on roughly half the molecules of AmesFormer-Honma, AmesFormer-Hansen is necessarily the most overparameterised relative to its dataset size. Consequently, it would be inclined to memorise the data, leading to longer convergence times and potentially poorer generalisation to unseen data (Zhang et al. 2016). The fact that the model with the largest dataset, AmesFormer-Duo converged the earliest (Figure 3.2c) lends further credence to this idea.

However, this cannot explain the worse performance of AmesFormer-Hansen as overparameterisation has been demonstrated to actually improve the performance NN models, despite slowing their

convergence (Allen-Zhu et al. 2018). Instead, the poorer performance of AmesFormer-Hansen likely stems from the more mundane explanation that the smaller Hansen dataset contains fewer examples that generalise well to the Honma test set. Thus, AmesFormer-Hansen may lack the diverse examples needed to build a robust predictive model. This suggests a difference in the applicability domains of the Hansen and Honma datasets.

Some limited evidence suggests that overparameterisation can hinder test-set performance by pushing the model to learn spurious correlations (Sagawa et al. 2020). In the case of AmesFormer, this would mean the network might learn that irrelevant features, which coincidentally appear in more mutagenic molecules but are not causally related to mutagenicity, are indicative of a positive result. This phenomenon could lead to poor generalization on unseen data. While these findings are primarily demonstrated in the realm of image-recognition models, they highlight the potential risks of overparameterisation in any complex model (Sagawa et al. 2020).

4.3 Understanding the Performance of AmesFormer

The two models superior to AmesFormer-Pro in the Furuhama et al. (2023) Ames prediction challenge, ChemTunes and MMI-STK2, are both ensemble models. Below, we discuss two primary advantages of ensembles and how they may explain their superiority to AmesFormer.

Ensemble models can self-mitigate their weaknesses. Ensemble models, which combine the predictions of multiple base models such as logistic regression, NN, and GNNs, leverage the strengths of each to enhance their overall performance. This process involves a voting mechanism where the individual model predictions are weighted and aggregated to produce a final, more accurate prediction. The power of ensemble methods lies in their ability to mitigate the weaknesses of individual models by combining them into a robust single learner, often leading to improved generalization and reduced overfitting in complex tasks (Dietterich 2000; Rokach 2010). Such techniques have demonstrated significant success across various domains, including bioinformatics, cheminformatics, and financial forecasting (Mienye and Sun 2022; Wang et al. 2023). Thus, the strength of ensembles lays in their ability to aggregate multiple weak learners into a single strong learner.

Ensembles access richer chemical information. In the cheminformatics context, ensembles may also have better access to molecular information compared to a NN or GNN alone. To illustrate this point for the Ames prediction task, consider an ensemble of a NN and a GNN. The NN operating on fingerprint data may access the imprecise atom and bond information encoded by the MF, and the global features encoded by some fingerprints (i.e., Mordred MF which encode LogP solubility information). As in AmesFormer the constituent GNN may still access the detailed atom-level and bond-level

information. Thus, the output of the ensemble considers both the precise information yielded by the GNN, and the imprecise whole-molecule information produced by the NN which processes MF. We hypothesise that the extra global molecular information contained within the MF fed to the constituent NN also contributes to the superiority of ensemble models in the Ames task. As AmesFormer is unable to access this MF-encoded information, and only considers atom and edge data, not whole-molecule information, it underperforms such ensemble models.

Despite these limitations, AmesFormer still performs relatively well, superior to many weaker ensemble models, such as the Leadscope Consensus Model (Table 3.2) (Furuhama et al. 2023). This seems counterintuitive, as ensembles should be more powerful learners with access to broader information. We propose two reasons why AmesFormer performs favourably against some, but not all, ensemble models.

AmesFormer is an independently strong learner. It is possible that AmesFormer is such a strong independent learner that it may perform better than many ensemble models and MF-based methods even without access to any global molecular features.

This is supported by many GNN benchmarks that show SOTA performance on cheminformatics tasks even without access to global molecular information. For example, Graphormer by Chengxuan et al. (2021) achieved the best performance to date in 2021 on the ZINC-500K molecular prediction task with an mean absolute error (MAE) of 0.122. However, in 2022, Rampášek et al. (2022), reported a ZINC-500K MAE of 0.070 by combining a GNN with whole-molecule MF data. Though more recently Menegaux et al. (2023) and Ma et al. (2023) achieved SOTA MAEs of 0.056 and 0.059, respectively, without using whole-molecule features. This supports the notion that GNNs are strong enough learners to achieve excellent performance even without whole-molecule information, such as solubility.

These empirical findings are supported by theoretical work which shows that GNNs are more expressive than MF-based NNs for molecular data (Huang and Villar 2022; Kanatsoulis and Ribeiro 2023). The lower resolution of MF fails to exhaustively capture all atomic and edge features of a molecule, making it possible for structurally similar but pharmacologically distinct molecules to generate identical representations. This may lead to incorrect mutagenicity predictions within traditional MF or NN frameworks.

Xu, Hu, et al. (2019) argue that this issue is unlikely with GNNs, as they can differentiate between any two graphs classified as non-isomorphic by the Weisfeiler-Lehman test (WL test) test. The WL test test iteratively updates node features $\mathbf{h}_i^{(k)}$ by aggregating the features of neighboring nodes $h_{uu\in N_i}$ until all labels across the graph stabilise eq. (4.1) (Leman 2018; Huang and Villar 2022). If the features of all nodes in a graph are non-identical after stabilisation, the WL test test considers the graphs non-isomorphic (Huang and Villar 2022). Essentially, GNNs perform the WL test test during their

AGGREGATE-UPDATE steps (Xu, Hu, et al. 2019). Note the similarity between the aggregation function used in GNNs eq. (1.17) and the WL test aggregation eq. (4.1). Therefore, even if two graphs are structurally similar, if their node features differ within any shared substructure between the molecules, they will be WL test non-isomorphic and hence produce divergent representations within the GNN. This capability allows GNNs to accurately distinguish between structurally similar but pharmacologically distinct molecules, thereby avoiding misclassification.

$$\mathbf{h}_i^{(k+1)} = \text{AGGREGATE}\left(\mathbf{h}_i^{(k)}, \{h_u\}_{u \in N_i}\right) \tag{4.1}$$

However, because the WL test test is locality-based, two structurally distinct graphs G and G' could yield identical multisets of node labels, causing the model to represent them identically eq. (4.2).

$$\{\mathbf{h}_i^{(k)} : i \in V(G)\} = \{\mathbf{h}_{i'}^{(k)} : i' \in V(G')\} \tag{4.2}$$

Fortunately, the SPD information incorporated into AmesFormer guards against this issue (Chengxuan et al. 2021). The SPD encoding captures the global structural information of the graph, enabling AmesFormerto distinguish between graphs with identical multisets of node labels but different overall structures (Chengxuan et al. 2021). This additional information allows AmesFormer to create unique representations for structurally distinct graphs, overcoming the limitations of earlier GNNs that relied solely on local neighborhood information. A proof for this is presented in Chengxuan et al. (2021). Thus, it is highly unlikely that the SPD-enhanced WL test test within GNNs could fail to distinguish and classify distinct molecules.

Additionally, recent research indicates that GNNs can differentiate any two graphs that differ in at least one eigenvalue of their graph Laplacians (Kanatsoulis and Ribeiro 2023). Thus as long as a single node feature differs between two molecular graphs, which must be the case if they have differing pharmacology, AmesFormer will always produce a correspondingly unique internal representation.

Additionally, recent research indicates that GNNs can distinguish between any two graphs that differ in at least one eigenvalue of their graph Laplacians (Kanatsoulis and Ribeiro 2023). The graph Laplacian $\mathbf{L}$ of a graph G is defined as:

$$\mathbf{L} = \mathbf{D} - \mathbf{A}, \tag{4.3}$$

where $\mathbf{D}$ is the degree matrix and $\mathbf{A}$ is the adjacency matrix. Two graphs G and G' are distinguished if their Laplacians have different eigenvalues:

$$\lambda_i(G) \neq \lambda_i(G') \text{ for some eigenvalue } \lambda_i. \tag{4.4}$$

Since it is impossible for two pharmacologically distinct molecules to produce graphs that both fail the SPD-enhanced WL test test and share identical spectral properties of their graph Laplacians, we believe the chance of AmesFormer generating identical representations for non-identical molecules, and thus misclassifying them, is essentially zero. This is a significant advantage of AmesFormer over MF-based NNs, which may incorrectly produce identical representations for distinct molecules, leading to misclassification. This superior expressivity may explain the superior performance of AmesFormer.

4.3.1 Future Directions: Improving AmesFormer

Node-edge ‘‘Cross-attention’’. In AmesFormer attention, each node feature attends to itself and all other node features, as does each edge feature. Let A_V, shown in Figure 4.1a, be the node-to-node attention matrix where each element v_{ij} represents the attention score from node i to node j. Analogously, let A_E, shown in Figure 4.1b, be edge-to-edge attention matrix where each element e_{ij} represents the attention score from edge i to node j. The total computational complexity of AmesFormer attention is $O(|V|^2) + O(|E|^2)$, where $|V|$ and $|E|$ denote the cardinality of the set of nodes V and edges E.

$$A_V = \begin{bmatrix} v_{11} & v_{12} & \cdots & v_{1n} \\ v_{21} & v_{22} & \cdots & v_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ v_{n1} & v_{n2} & \cdots & v_{nn} \end{bmatrix} \qquad A_E = \begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ e_{n1} & e_{n2} & \cdots & e_{nn} \end{bmatrix}$$

(a) $V \times V$ matrix of node-to-node attention coefficients. (b) $E \times E$ matrix of edge-to-edge attention coefficients.

Figure 4.1: The attention matrices currently implemented in AmesFormer.

There is no node-edge ‘‘cross-attention’’ as node features cannot attend to edge features or vice versa. This is a missed opportunity, as one can intuitively understand that atomic features may influence, or be influenced by, the bonds in which they participate.

Straightforwardly, we may implement this ‘‘cross-attention’’ by creating a diagonal block matrix of A_V and A_E Figure 4.2. Within this matrix, the node-node $V \times V$ and edge-edge $E \times E$ portions would be equivalent to existing AmesFormer attention, whilst the new information contained within the $V \times E$ and $E \times V$ represents new ‘‘cross-attention’’ information. We hypothesise that such cross-attention would improve the performance of AmesFormer for relatively moderate computational cost; an increase from from $O(|V|^2 + |E|^2)$ to $O((|V| + |E|)^2)$.

Implementation of this node-edge ‘‘cross attention’’ is an open path toward algorithmic improvement.

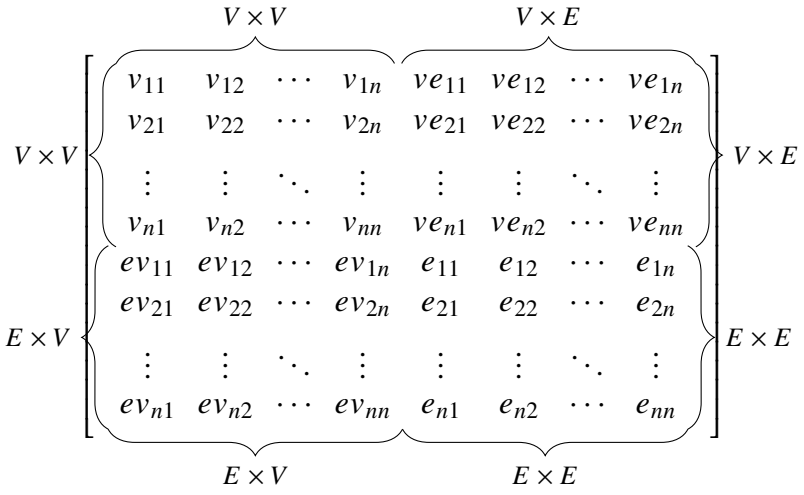
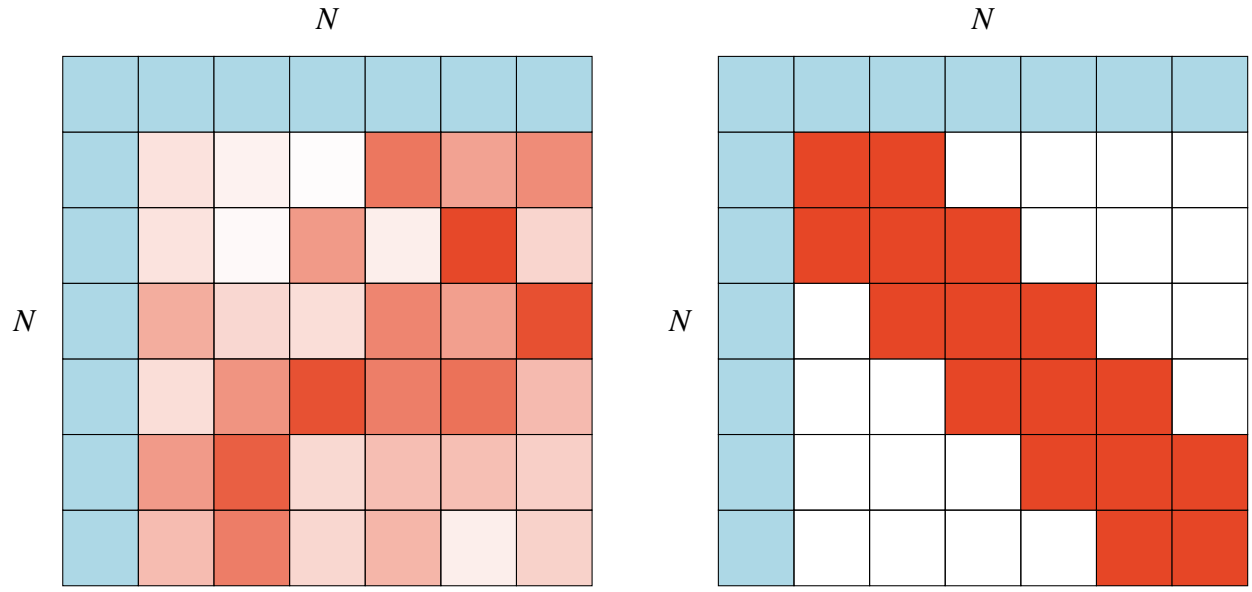


Figure 4.2: A possible arrangement of the node-edge cross-attention coefficient matrix. The $V \times V$ and $E \times E$ matrices are equivalent to the matrices presented in Figure 4.1a and Figure 4.1b, respectively. The new $V \times E$ and $E \times V$ matrices represent cross attention; information newly available to the model. For example ve_{11} denotes cross-attention from node v_1 to edge e_1.

The Possibilities of Efficient Attention. The computational complexity of training represents a major challenge for more broadly applying AmesFormer. Despite our small dataset (n = 7 163), model training still took approximately 6 hours. Whilst manageable for our task, many other graph tasks, such as those featured in Hu et al. (2020), use datasets of up to 3.7 million molecules.

Our inefficiency is largely a result of the full dot-product self-attention layers within transformers, which are responsible for both their great expressiveness, and undesirable $O(N^2)$ computational complexity. This quadratic complexity arises as each input token (node feature) $n \in N$ attends to itself and all other tokens. Hence, for any input of length N, self-attention requires $N \times N$ computations. For academic researchers, who are generally unable to leverage massive compute clusters, this restricts the training of large transformers and hence limits their access to SOTA techniques. Attention mechanisms with of $O(N)$ or $O(N \log N)$ complexity, such as Ring Attention and Big Bird, are promising strategies to ameliorate the quadratic complexity of transformers (Liu, Zaharia, et al. 2023; Zaheer et al. 2020). Efficient attention mechanisms usually calculate the attention coefficients for proximal nodes, and only occasionally calculate attention for non-local node pairs (Zaheer et al. 2020). This exploits the locality bias of much data, wherein proximal tokens are often more important (Zaheer et al. 2020). Hence, one may drop the calculation of most non-local coefficients as they are usually unimportant, saving computation with little performance impact (Lin et al. 2021).

Applied to the $N \times N$ node attention matrices of a GTN, instead of calculating all attention coefficients between node pairs as in Figure 4.3a, efficient attention would only calculate them between a subset of node feature pairs as in Figure 4.3b. An alternative strategy is FiSH attention, wherein only k full



(a) Conventional scaled dot product self-attention as implemented in AmesFormer (Vaswani et al. 2017). Note that coefficients are calculated for all node pairs. For distant nodes, which rarely need to attend eachother and hence have negligable coefficients, this is wasted computation.

(b) Big Bird attention (Zaheer et al. 2020; Lin et al. 2021), a more efficient alternative to the scaled dot product self-attention of AmesFormer. Proximal nodes are always attended, but only a subset of non-proximal pairs have their coefficients calculated, saving computation.

Figure 4.3: A visualisation of the node-to-node attention coefficients between two attention mechanisms. Brighter orange denotes greater attention between those two nodes.

attention heads are learned, and the remaining heads are calculated as a learned linear combination of those k global heads (Nguyen et al. 2022). However, this has never been demonstrated in a GTN.

Application of these efficient attention mechanisms to AmesFormer would greatly improve computational efficiency of model training, and hence democratise access to the training of SOTA graph transformers. A number of well-performing linear attention GTN have emerged recently using local attention (akin to Figure 4.3b) and attention kernel decomposition (Wu, Xu, et al. 2023; Deng et al. 2024). However, none of these strategies have been applied to biochemical data as of May 2024 (Deng et al. 2024). As such, integration of efficient attention mechanisms with cheminformatics GTNs represents a promising and open research direction.

In conclusion, we present AmesFormer, a novel molecular graph transformer neural network architecture which has not been previously applied to Ames mutagenicity prediction. AmesFormer produces SOTA non-ensemble performance and represents an exciting addition to the regulatory arsenal for the control of community exposure to potentially toxic drug impurities and industrial chemicals.

AICIS has recently expressed interest in AmesFormer models and we plan to deploy it on our laboratory website.

Bibliography

Abdi, Hervé (2010). ‘‘Holm’s sequential Bonferroni procedure’’. In: *Encyclopedia of research design* 1.8. Publisher: Thousand Oaks, California, pp. 1–8.

AbdulHameed, Mohamed Diwan M., Ruifeng Liu, and Anders Wallqvist (2023). ‘‘Using a Graph Convolutional Neural Network Model to Identify Bile Salt Export Pump Inhibitors’’. In: *ACS omega* 8.24, pp. 21853–21861. ISSN: 2470-1343. DOI: 10.1021/acsomega.3c01583.

AICIS (2022). *Guide to categorising your chemical importation and manufacture - Step 4.4 Work out your human health hazard characteristics*. URL: https://www.industrialchemicals.gov.au/guide-categorising-your-chemical-importation-and-manufacture/step-4-work-out-your-introductions-risk-human-health/step-44-work-out-your-human-health-hazard-characteristics.

Akiba, Takuya et al. (July 2019). ‘‘Optuna: A Next-generation Hyperparameter Optimization Framework’’. In: *arXiv e-prints*. _eprint: 1907.10902, arXiv:1907.10902. DOI: 10.48550/arXiv.1907.10902.

Allen-Zhu, Zeyuan, Yuanzhi Li, and Yingyu Liang (Nov. 2018). ‘‘Learning and Generalization in Overparameterized Neural Networks, Going Beyond Two Layers’’. In: *arXiv e-prints*. _eprint: 1811.04918, arXiv:1811.04918. DOI: 10.48550/arXiv.1811.04918.

Altman, Naomi and Martin Krzywinski (2018). ‘‘The curse(s) of dimensionality’’. In: *Nature methods* 15.6, pp. 399–400. ISSN: 1548-7091. DOI: 10.1038/s41592-018-0019-x.

Ames, B. N., F. D. Lee, and W. E. Durston (Mar. 1973). ‘‘An improved bacterial test system for the detection and classification of mutagens and carcinogens.’’ In: *Proceedings of the National Academy of Sciences of the United States of America* 70.3. Place: United States, pp. 782–786. ISSN: 0027-8424 1091-6490. DOI: 10.1073/pnas.70.3.782.

Ashby, John et al. (1993). ‘‘Mechanistic Relationship among Mutagenicity, Skin Sensitization, and Skin Carcinogenicity’’. In: *Environmental health perspectives* 101.1, pp. 62–67. ISSN: 0091-6765. DOI: 10.1289/ehp.9310162.

Baxter, J. (June 1, 2011). ‘‘A Model of Inductive Bias Learning’’. In: arXiv:1106.0245. DOI: 10.48550/arXiv.1106.0245.

Bishnoi, Suresh et al. (2022). ‘‘Enhancing the Inductive Biases of Graph Neural ODE for Modeling Dynamical Systems’’. In: *arXiv.org*. ISSN: 2331-8422. DOI: 10.48550/arxiv.2209.10740.

Bolton, Evan E. et al. (Jan. 1, 2008). ‘‘Chapter 12 - PubChem: Integrated Platform of Small Molecules and Biological Activities’’. In: *Annual Reports in Computational Chemistry*. Ed. by Ralph A. Wheeler and David C. Spellmeyer. Vol. 4. Elsevier, pp. 217–241. ISBN: 1574-1400.

Brodersen, Kay H et al. (2010). ‘‘The Balanced Accuracy and Its Posterior Distribution’’. In: *2010 20th International Conference on Pattern Recognition*. ISSN: 1051-4651. IEEE, pp. 3121–3124. ISBN: 1-4244-7542-2.

Brown, Tom B. et al. (May 2020). ‘‘Language Models are Few-Shot Learners’’. In: *arXiv e-prints*. _eprint: 2005.14165, arXiv:2005.14165. DOI: 10.48550/arXiv.2005.14165.

Capecchi, Alice, Daniel Probst, and Jean-Louis Reymond (June 12, 2020). ‘‘One molecular fingerprint to rule them all: drugs, biomolecules, and the metabolome’’. In: *Journal of Cheminformatics* 12.1, p. 43. ISSN: 1758-2946. DOI: 10.1186/s13321-020-00445-4.

Chengxuan, Ying et al. (2021). ‘‘Do Transformers Really Perform Bad for Graph Representation?’’ In: *arXiv.org*. ISSN: 2331-8422. DOI: 10.48550/arxiv.2106.05234.

Conover, W. J. and R. L. Iman (1979). *Multiple-comparisons procedures. Informal report*. United States, Medium: ED, Size: Pages: 17.

Demsar, J. (2006). ‘‘Statistical comparisons of classifiers over multiple data sets’’. In: *Journal of machine learning research* 7, pp. 1–30. ISSN: 1533-7928.

Deng, Chenhui, Zichao Yue, and Zhiru Zhang (2024). ‘‘Polynormer: Polynomial-Expressive Graph Transformer in Linear Time’’. In: *The Twelfth International Conference on Learning Representations*. URL: https://openreview.net/forum?id=hmv1LpNfXa.

Devlin, Jacob et al. (Oct. 2018). ‘‘BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding’’. In: *arXiv e-prints*. _eprint: 1810.04805, arXiv:1810.04805. DOI: 10.48550/arXiv.1810.04805.

Dietterich, Thomas G. (2000). ‘‘Ensemble Methods in Machine Learning’’. In: *Multiple Classifier Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 1–15. ISBN: 978-3-540-45014-6.

Dijkstra, Edsger W (1959). ‘‘A note on two problems in connexion with graphs’’. In: *Numerische mathematik* 1.1. Publisher: Springer, pp. 269–271.

Durant, Joseph L. et al. (2002). ‘‘Reoptimization of MDL Keys for Use in Drug Discovery’’. In: *Journal of Chemical Information and Computer Sciences* 42.6, pp. 1273–1280. ISSN: 0095-2338. DOI: 10.1021/ci010132r.

Duval, Alexandre et al. (Dec. 1, 2023). ‘‘A Hitchhiker’s Guide to Geometric GNNs for 3D Atomic Systems’’. In: arXiv:2312.07511. DOI: 10.48550/arXiv.2312.07511.

Elnaggar, Ahmed et al. (July 2020). ‘‘ProtTrans: Towards Cracking the Language of Life’s Code Through Self-Supervised Deep Learning and High Performance Computing’’. In: *arXiv e-prints*. _eprint: 2007.06225, arXiv:2007.06225. DOI: 10.48550/arXiv.2007.06225.

European Communities (2006). *Regulation (EC) No 1907/2006 of the European Parliament and of the Council of 18 December 2006 concerning the Registration, Evaluation, Authorisation and Restriction of Chemicals (REACH), establishing a European Chemicals Agency, amending Directive 1999/45/EC and repealing Council Regulation (EEC) No 793/93 and Commission Regulation (EC) No 1488/94 as well as Council Directive 76/769/EEC and Commission Directives 91/155/EEC, 93/67/EEC, 93/105/EC and 2000/21/EC*, pp. 1–849.

Fey, Matthias and Jan Eric Lenssen (Mar. 1, 2019). ‘‘Fast Graph Representation Learning with PyTorch Geometric’’. In: *arXiv e-prints*, arXiv:1903.02428. DOI: 10.48550/arXiv.1903.02428. URL: https://ui.adsabs.harvard.edu/abs/2019arXiv190302428F.

Flach, Peter and Edson Matsubara (Jan. 2007). ‘‘On classification, ranking, and probability estimation.’’ In.

Floyd, Robert W. (June 1962). ‘‘Algorithm 97: Shortest path’’. In: *Commun. ACM* 5.6. Place: New York, NY, USA Publisher: Association for Computing Machinery, p. 345. ISSN: 0001-0782. DOI: 10.1145/367766.368168. URL: https://doi.org/10.1145/367766.368168.

François Fleuret (Sept. 20, 2023). *Little Book of Deep Learning*. URL: https://fleuret.org/public/lbdl.pdf.

Furuhama, A. et al. (2023). ‘‘Evaluation of QSAR models for predicting mutagenicity: outcome of the Second Ames/QSAR international challenge project’’. In: *SAR and QSAR in environmental research* 34.12, pp. 983–1001. ISSN: 1062-936X. DOI: 10.1080/1062936X.2023.2284902.

Gal, Yarin and Zoubin Ghahramani (June 20, 2016). ‘‘Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning’’. In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, pp. 1050–1059. URL: https://proceedings.mlr.press/v48/gal16.html.

Gao, Jian et al. (2023). ‘‘TransFoxMol: predicting molecular property with focused attention’’. In: *Briefings in Bioinformatics* 24.5. ISSN: 1477-4054. DOI: 10.1093/bib/bbad306.

Gasteiger, Johannes, Florian Becker, and Stephan Günnemann (June 1, 2021). ‘‘GemNet: Universal Directional Graph Neural Networks for Molecules’’. In: arXiv:2106.08903. DOI: 10.48550/arXiv.2106.08903.

Glorot, Xavier et al. (2010). ‘‘Understanding the difficulty of training deep feedforward neural networks’’. Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics. Proceedings of Machine Learning Research.

Guo, Chuan et al. (June 2017). ‘‘On Calibration of Modern Neural Networks’’. In: *arXiv e-prints*. _eprint: 1706.04599, arXiv:1706.04599. DOI: 10.48550/arXiv.1706.04599.

Al-hammuri, Khalid et al. (July 10, 2023). ‘‘Vision transformer architecture and applications in digital health: a tutorial and survey’’. In: *Visual Computing for Industry, Biomedicine, and Art* 6.1, p. 14. ISSN: 2524-4442. DOI: 10.1186/s42492-023-00140-9. URL: https://doi.org/10.1186/s42492-023-00140-9.

Han, Jason J. (Mar. 2023). ‘‘FDA Modernization Act 2.0 allows for alternatives to animal testing.’’ In: *Artificial organs* 47.3. Place: United States, pp. 449–450. ISSN: 1525-1594 0160-564X. DOI: 10.1111/aor.14503.

Hansen, Katja et al. (2009). ‘‘Benchmark data set for in silico prediction of Ames mutagenicity’’. In: *Journal of chemical information and modeling* 49.9, p. 2077. ISSN: 1549-960X. DOI: 10.1021/ci900161g.

Hartenfeller, Markus and Gisbert Schneider (2011). ‘‘Enabling future drug discovery by de novo design’’. In: *Wiley interdisciplinary reviews. Computational molecular science* 1.5. Place: Hoboken, USA Publisher: John Wiley & Sons, Inc, pp. 742–759. ISSN: 1759-0876. DOI: 10.1002/wcms.49.

He, Kaiming et al. (Feb. 1, 2015). ‘‘Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification’’. In: *arXiv e-prints*, arXiv:1502.01852. DOI: 10.48550/arXiv.1502.01852. URL: https://ui.adsabs.harvard.edu/abs/2015arXiv150201852H.

Hendrycks, Dan and Kevin Gimpel (June 2016). ‘‘Gaussian Error Linear Units (GELUs)’’. In: *arXiv e-prints*. _eprint: 1606.08415, arXiv:1606.08415. DOI: 10.48550/arXiv.1606.08415.

Hernández, Saiveth and Pedro J. Ballester (Mar. 8, 2023). ‘‘On the Best Way to Cluster NCI-60 Molecules.’’ In: *Biomolecules* 13.3. Place: Switzerland. ISSN: 2218-273X. DOI: 10.3390/biom13030498.

Hicks, Steven A. et al. (Apr. 8, 2022). ‘‘On evaluation metrics for medical applications of artificial intelligence.’’ In: *Scientific reports* 12.1. Place: England, p. 5979. ISSN: 2045-2322. DOI: 10.1038/s41598-022-09954-8.

Holm, Sture (1979). ‘‘A simple sequentially rejective multiple test procedure’’. In: *Scandinavian journal of statistics*. Publisher: JSTOR, pp. 65–70.

Honma, Masamitsu et al. (2019). ‘‘Improvement of quantitative structure–activity relationship (QSAR) tools for predicting Ames mutagenicity: outcomes of the Ames/QSAR International Challenge Project’’. In: *Mutagenesis* 34.1. Place: UK Publisher: Oxford University Press, pp. 3–16. ISSN: 0267-8357. DOI: 10.1093/mutage/gey031.

Hosseini, M., D. J. Maddalena, and I. Spence (Nov. 1, 1997). ‘‘Using Artificial Neural Networks To Classify the Activity of Capsaicin and Its Analogues’’. In: *Journal of Chemical Information and Computer Sciences* 37.6. Publisher: American Chemical Society, pp. 1129–1137. ISSN: 0095-2338. DOI: 10.1021/ci9700384. URL: https://doi.org/10.1021/ci9700384.

Hu, Weihua et al. (May 2020). ‘‘Open Graph Benchmark: Datasets for Machine Learning on Graphs’’. In: *arXiv e-prints*. _eprint: 2005.00687, arXiv:2005.00687. DOI: 10.48550/arXiv.2005.00687.

Huang, Ningyuan and Soledad Villar (Jan. 2022). ‘‘A Short Tutorial on The Weisfeiler-Lehman Test And Its Variants’’. In: *arXiv e-prints*. _eprint: 2201.07083, arXiv:2201.07083. DOI: 10.48550/arXiv.2201.07083.

Hung, Chiakang and Giuseppina Gini (2021). ‘‘QSAR modeling without descriptors using graph convolutional neural networks: the case of mutagenicity prediction’’. In: *Molecular diversity* 25.3, pp. 1283–1299. ISSN: 1381-1991. DOI: 10.1007/s11030-021-10250-2.

ICH (2013). *ICH guideline S2 (R1) on genotoxicity testing and data interpretation for pharmaceuticals intended for human use*. ICH.

— (2017). *ASSESSMENT AND CONTROL OF DNA REACTIVE (MUTAGENIC) IMPURITIES IN PHARMACEUTICALS TO LIMIT POTENTIAL CARCINOGENIC RISK*. ICH.

Jin, Jiarui et al. (2022). ‘‘Refined Edge Usage of Graph Neural Networks for Edge Prediction’’. In: *arXiv.org*. ISSN: 2331-8422. DOI: 10.48550/arxiv.2212.12970.

Kamber, Markus et al. (May 2009). ‘‘Comparison of the Ames II and traditional Ames test responses with respect to mutagenicity, strain specificities, need for metabolism and correlation with rodent carcinogenicity’’. In: *Mutagenesis* 24.4. _eprint: https://academic.oup.com/mutage/article-pdf/24/4/359/3787533/gep017.pdf, pp. 359–366. ISSN: 0267-8357. DOI: 10.1093/mutage/gep017. URL: https://doi.org/10.1093/mutage/gep017.

Kanatsoulis, Charilaos and Alejandro Ribeiro (2023). ‘‘Graph Neural Networks Are More Powerful Than we Think’’. In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=lgYzzQ0fX5D.

Kearnes, Steven et al. (Aug. 2016). ‘‘Molecular graph convolutions: moving beyond fingerprints’’. In: *Journal of Computer-Aided Molecular Design* 30.8. _eprint: 1603.00856, pp. 595–608. DOI: 10.1007/s10822-016-9938-8.

Keogh, Eamonn and Abdullah Mueen (2010). ‘‘Curse of Dimensionality’’. In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, pp. 257–258. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_192. URL: https://doi.org/10.1007/978-0-387-30164-8_192.

Kingma, Diederik P. and Jimmy Ba (Dec. 2014). ‘‘Adam: A Method for Stochastic Optimization’’. In: *arXiv e-prints*. _eprint: 1412.6980, arXiv:1412.6980. DOI: 10.48550/arXiv.1412.6980.

Kong, Xiaorui et al. (2023). ‘‘Machine Learning Techniques Applied to the Study of Drug Transporters’’. In: *Molecules (Basel, Switzerland)* 28.16. Place: Basel Publisher: MDPI AG, pp. 5936–. ISSN: 1420-3049.

Krizhevsky, Alex, Ilya Sutskever, and Geoffrey Hinton (2017). *ImageNet classification with deep convolutional neural networks*. ISSN: 0001-0782 Issue: 6 Pages: 84-90 Place: New York Publication Title: Communications of the ACM Volume: 60.

Kumar, Rajnish et al. (Sept. 1, 2021). ‘‘A deep neural network–based approach for prediction of mutagenicity of compounds’’. In: *Environmental Science and Pollution Research* 28.34, pp. 47641–47650. ISSN: 1614-7499. DOI: 10.1007/s11356-021-14028-9. URL: https://doi.org/10.1007/s11356-021-14028-9.

Landrum, Greg (2012). *RDKit - MACCS Keys Implementation*.

Leman, Adrien (2018). ‘‘THE REDUCTION OF A GRAPH TO CANONICAL FORM AND THE ALGEBRA WHICH APPEARS THEREIN’’. In: URL: https://api.semanticscholar.org/CorpusID:49579538.

Li, Lisha, Kevin Jamieson, et al. (Mar. 2016). ‘‘Hyperband: A Novel Bandit-Based Approach to Hyperparameter Optimization’’. In: *arXiv e-prints*. _eprint: 1603.06560, arXiv:1603.06560. DOI: 10.48550/arXiv.1603.06560.

Li, Ting, Zhichao Liu, et al. (Oct. 2023). ‘‘DeepAmes: A deep learning-powered Ames test predictive model with potential for regulatory application.’’ In: *Regulatory toxicology and pharmacology : RTP* 144. Place: Netherlands, p. 105486. ISSN: 1096-0295 0273-2300. DOI: 10.1016/j.yrtph.2023.105486.

Lin, Tianyang et al. (June 2021). ‘‘A Survey of Transformers’’. In: *arXiv e-prints*. _eprint: 2106.04554, arXiv:2106.04554. DOI: 10.48550/arXiv.2106.04554.

Liu, Hao, Matei Zaharia, and Pieter Abbeel (Oct. 1, 2023). ‘‘Ring Attention with Blockwise Transformers for Near-Infinite Context’’. In: arXiv:2310.01889. DOI: 10.48550/arXiv.2310.01889.

Liu, Jie, Grace Patlewicz, et al. (Nov. 20, 2017). ‘‘Predicting Organ Toxicity Using in Vitro Bioactivity Data and Chemical Structure.’’ In: *Chemical research in toxicology* 30.11. Place: United States, pp. 2046–2059. ISSN: 1520-5010 0893-228X. DOI: 10.1021/acs.chemrestox.7b00084.

Loshchilov, Ilya and Frank Hutter (Nov. 2017). ‘‘Decoupled Weight Decay Regularization’’. In: *arXiv e-prints*. _eprint: 1711.05101, arXiv:1711.05101. DOI: 10.48550/arXiv.1711.05101.

Luchini, Guilian (2021). ‘‘Fingerprints: Clashing and Clustering’’. In.

Lui, Raymond, Davy Guan, and Slade Matthews (2023). ''Mechanistic Task Groupings Enhance Multitask Deep Learning of Strain-Specific Ames Mutagenicity''. In: *Chemical research in toxicology* 36.8, pp. 1248–1254. ISSN: 0893-228X. DOI: 10.1021/acs.chemrestox.2c00385.

Ma, Liheng et al. (May 2023). ''Graph Inductive Biases in Transformers without Message Passing''. In: *arXiv e-prints*. _eprint: 2305.17589, arXiv:2305.17589. DOI: 10.48550/arXiv.2305.17589.

Maron, Dorothy M. and Bruce N. Ames (1983). ''Revised methods for the Salmonella mutagenicity test''. In: *Mutation Research/Environmental Mutagenesis and Related Subjects* 113.3, pp. 173–215. ISSN: 0165-1161. DOI: 10.1016/0165-1161(83)90010-9.

McInnes, Leland, John Healy, and James Melville (Feb. 2018). ''UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction''. In: *arXiv e-prints*. _eprint: 1802.03426, arXiv:1802.03426. DOI: 10.48550/arXiv.1802.03426.

Menegaux, Romain et al. (Apr. 2023). ''Self-Attention in Colors: Another Take on Encoding Graph Structure in Transformers''. In: *arXiv e-prints*. _eprint: 2304.10933, arXiv:2304.10933. DOI: 10.48550/arXiv.2304.10933.

Mienye, Ibomoiye Domor and Yanxia Sun (2022). ''A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects''. In: *IEEE Access* 10, pp. 99129–99149. DOI: 10.1109/ACCESS.2022.3207287.

Nguyen, Tan et al. (2022). ''Improving Transformer with an Admixture of Attention Heads''. In: *Advances in Neural Information Processing Systems*. Ed. by S. Koyejo et al. Vol. 35. Curran Associates, Inc., pp. 27937–27952. URL: https://proceedings.neurips.cc/paper_files/paper/2022/file/b2e4edd53059e24002a0c916d75cc9a3-Paper-Conference.pdf.

Niculescu-Mizil, Alexandru and Rich Caruana (2005). ''Predicting good probabilities with supervised learning''. In: *Proceedings of the 22nd international conference on Machine learning*, pp. 625–632.

Noutahi, Emmanuel et al. (Sept. 2023). *Datamol*. Version 0.9.4. DOI: 10.5281/zenodo.8373019. URL: https://doi.org/10.5281/zenodo.8373019.

OECD (2014). *Guidance Document on the Validation of (Quantitative) Structure-Activity Relationship [(Q)SAR] Models*. Type: doi:https://doi.org/10.1787/9789264085442-en. URL: https://www.oecd-ilibrary.org/content/publication/9789264085442-en.

— (2020). *Overview of Concepts and Available Guidance related to Integrated Approaches to Testing and Assessment (IATA)*. 329. Paris, France. URL: https://www.oecd.org/chemicalsafety/risk-assessment/concepts-and-available-guidance-related-to-integrated-approaches-to-testing-and-assessment.pdf.

Parmar, Niki et al. (Feb. 2018). ''Image Transformer''. In: *arXiv e-prints*. _eprint: 1802.05751, arXiv:1802.05751. DOI: 10.48550/arXiv.1802.05751.

Paszke, Adam et al. (Dec. 1, 2019). ‘‘PyTorch: An Imperative Style, High-Performance Deep Learning Library’’. In: *arXiv e-prints*, arXiv:1912.01703. DOI: 10.48550/arXiv.1912.01703. URL: https://ui.adsabs.harvard.edu/abs/2019arXiv191201703P.

Patlewicz, G. et al. (2010). ‘‘Can mutagenicity information be useful in an Integrated Testing Strategy (ITS) for skin sensitization?’’ In: *SAR and QSAR in environmental research* 21.7, pp. 619–656. ISSN: 1062-936X. DOI: 10.1080/1062936X.2010.528447.

Paykan Heyrati, Mehdi et al. (Nov. 28, 2023). ‘‘BioAct-Het: A Heterogeneous Siamese Neural Network for Bioactivity Prediction Using Novel Bioactivity Representation’’. In: *ACS omega* 8.47, pp. 44757–44772. DOI: 10.1021/acsomega.3c05778.

Provost, Foster, Tom Fawcett, and Ron Kohavi (Apr. 2001). ‘‘The Case Against Accuracy Estimation for Comparing Induction Algorithms’’. In: *Proceedings of the Fifteenth International Conference on Machine Learning*.

Radford, Alec et al. (Feb. 2021). ‘‘Learning Transferable Visual Models From Natural Language Supervision’’. In: *arXiv e-prints*. _eprint: 2103.00020, arXiv:2103.00020. DOI: 10.48550/arXiv.2103.00020.

Rampášek, Ladislav et al. (May 2022). ‘‘Recipe for a General, Powerful, Scalable Graph Transformer’’. In: *arXiv e-prints*. _eprint: 2205.12454, arXiv:2205.12454. DOI: 10.48550/arXiv.2205.12454.

Rocks, Jason W. and Pankaj Mehta (Aug. 2022). ‘‘Bias-variance decomposition of overparameterized regression with random linear features’’. In: *pre* 106.2. _eprint: 2203.05443, p. 025304. DOI: 10.1103/PhysRevE.106.025304.

Rogers, David and Mathew Hahn (May 24, 2010). ‘‘Extended-Connectivity Fingerprints’’. In: *Journal of chemical information and modeling* 50.5, pp. 742–754. ISSN: 1549-9596. DOI: 10.1021/ci100050t.

Rogers, David J. and Taffee T. Tanimoto (Oct. 1, 1960). ‘‘A Computer Program for Classifying Plants’’. In: *Science* 132, pp. 1115–1118. ISSN: 0036-8075. DOI: 10.1126/science.132.3434.1115.

Rokach, Lior (Feb. 1, 2010). ‘‘Ensemble-based classifiers’’. In: *Artificial Intelligence Review* 33.1, pp. 1–39. ISSN: 1573-7462. DOI: 10.1007/s10462-009-9124-7. URL: https://doi.org/10.1007/s10462-009-9124-7.

Rousseeuw, Peter J. (1987). ‘‘Silhouettes: A graphical aid to the interpretation and validation of cluster analysis’’. In: *Journal of Computational and Applied Mathematics* 20, pp. 53–65. ISSN: 0377-0427. DOI: https://doi.org/10.1016/0377-0427(87)90125-7. URL: https://www.sciencedirect.com/science/article/pii/0377042787901257.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (Oct. 1, 1986). ''Learning representations by back-propagating errors''. In: *Nature* 323.6088, pp. 533–536. ISSN: 1476-4687. DOI: 10.1038/323533a0.

Sagawa, Shiori et al. (May 2020). ''An Investigation of Why Overparameterization Exacerbates Spurious Correlations''. In: *arXiv e-prints*. _eprint: 2005.04345, arXiv:2005.04345. DOI: 10.48550/arXiv.2005.04345.

Scarselli, F. et al. (2009). ''The Graph Neural Network Model''. In: *IEEE transactions on neural networks* 20.1, pp. 61–80. ISSN: 1045-9227. DOI: 10.1109/TNN.2008.2005605.

Seo, Myungwon et al. (Jan. 22, 2020). ''Development of Natural Compound Molecular Fingerprint (NCMFP) with the Dictionary of Natural Products (DNP) for natural product-based drug development''. In: *Journal of Cheminformatics* 12.1, p. 6. ISSN: 1758-2946. DOI: 10.1186/s13321-020-0410-3.

Subramanian, Shreyas and Vignesh Ganapathiraman (2023). ''Zeroth order GreedyLR: An adaptive learning rate scheduler for deep neural network training''. In: *PRML 2023*. URL: https://www.amazon.science/publications/zeroth-order-greedylr-an-adaptive-learning-rate-scheduler-for-deep-neural-network-training.

Tennant, Raymond W. and John Ashby (1991). ''Classification according to chemical structure, mutagenicity to Salmonella and level of carcinogenicity of a further 39 chemicals tested for carcinogenicity by the U.S. National Toxicology Program''. In: *Mutation research. Reviews in genetic toxicology* 257.3. Place: Amsterdam Publisher: Elsevier B.V, pp. 209–227. ISSN: 0165-1110.

Vaswani, Ashish et al. (2017). ''Attention Is All You Need''. In: *arXiv.org*. ISSN: 2331-8422. DOI: 10.48550/arxiv.1706.03762.

Veličković, Petar et al. (2017). ''Graph Attention Networks''. In: Cornell University Library, arXiv.org. ISBN: 2331-8422. DOI: 10.48550/arxiv.1710.10903.

Wang, Binyou et al. (Sept. 12, 2023). ''Prediction of the effects of small molecules on the gut microbiome using machine learning method integrating with optimal molecular features.'' In: *BMC bioinformatics* 24.1. Place: England, p. 338. ISSN: 1471-2105. DOI: 10.1186/s12859-023-05455-1.

Ward, Joe H. (1963). ''Hierarchical Grouping to Optimize an Objective Function''. In: *Journal of the American Statistical Association* 58.301. Publisher: Taylor & Francis _eprint: https://www.tandfonline.com/doi/pdf/10.1080/01621459.1963.10500845, pp. 236–244. DOI: 10.1080/01621459.1963.10500845. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1963.10500845.

Wen, Wei et al. (Aug. 1, 2016). ''Learning Structured Sparsity in Deep Neural Networks''. In: arXiv:1608.03665. DOI: 10.48550/arXiv.1608.03665.

Wu, Wenxuan, Jiayu Qian, et al. (2023). ‘‘GeoDILI: A Robust and Interpretable Model for Drug-Induced Liver Injury Prediction Using Graph Neural Network-Based Molecular Geometric Representation’’. In: *Chemical research in toxicology*. ISSN: 0893-228X. DOI: 10.1021/acs.chemrestox.3c00199.

Wu, Yi, Yanyang Xu, et al. (2023). ‘‘KDLGT: a linear graph transformer framework via kernel decomposition approach’’. In: *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*. IJCAI ’23. Place: , Macao, P.R.China, ISBN: 978-1-956792-03-4. DOI: 10.24963/ijcai.2023/263. URL: https://doi.org/10.24963/ijcai.2023/263.

Xiong, Ruibin, Yunchang Yang, et al. (Feb. 2020). ‘‘On Layer Normalization in the Transformer Architecture’’. In: *arXiv e-prints*. _eprint: 2002.04745, arXiv:2002.04745. DOI: 10.48550/arXiv.2002.04745.

Xiong, Zhaoping, Dingyan Wang, et al. (Aug. 27, 2020). ‘‘Pushing the Boundaries of Molecular Representation for Drug Discovery with the Graph Attention Mechanism’’. In: *Journal of Medicinal Chemistry* 63.16, pp. 8749–8760. ISSN: 0022-2623. DOI: 10.1021/acs.jmedchem.9b00959.

Xu, Congying, Feixiong Cheng, et al. (2012). ‘‘In silico Prediction of Chemical Ames Mutagenicity’’. In: *Journal of chemical information and modeling* 52.11. Place: Washington, DC Publisher: American Chemical Society, pp. 2840–2847. ISSN: 1549-9596.

Xu, Keyulu, Weihua Hu, et al. (2019). ‘‘How powerful are graph neural networks?’’ In: *International Conference on Learning Representations*.

Xu, Youjun, Ziwei Dai, et al. (2015). ‘‘Deep Learning for Drug-Induced Liver Injury’’. In: *Journal of chemical information and modeling* 55.10, pp. 2085–2093. ISSN: 1549-9596. DOI: 10.1021/acs.jcim.5b00238.

Zaheer, Manzil et al. (July 1, 2020). ‘‘Big Bird: Transformers for Longer Sequences’’. In: arXiv:2007.14062. DOI: 10.48550/arXiv.2007.14062.

Zhang, Chiyuan et al. (Nov. 2016). ‘‘Understanding deep learning requires rethinking generalization’’. In: *arXiv e-prints*. _eprint: 1611.03530, arXiv:1611.03530. DOI: 10.48550/arXiv.1611.03530.

Zhong, Shifa and Xiaohong Guan (July 5, 2023). ‘‘Count-Based Morgan Fingerprint: A More Efficient and Interpretable Molecular Representation in Developing Machine Learning-Based Predictive Regression Models for Water Contaminants’ Activities and Properties’’. In: *Environmental Science & Technology*. ISSN: 0013-936X. DOI: 10.1021/acs.est.3c02198.

Glossary

bit collision When structurally similar chemical substructures are hashed into the same index of a MF bit vector. 9

graph A graph $G = (V, E)$ models entities as a V set of v nodes, and a E set of pairwise entity relationships e, known as edges, in non-Euclidean space. 16

Lipinski Limits Molecules with =< 5 Hydrogen Bond (H-bond) donors, =< 10 H-bond acceptors, molecular mass < 500 Da, Clog p < 5. 7

Acronyms

AICIS the Australian Industrial Chemicals Introduction Scheme 7, 18, 51

ANOVA analysis of variance 34

BA balanced accuracy 31–34, 38, 39

BCE Loss binary cross entropy loss 29, 30, 37

BSEP bile salt export pump 14, 18

CI confidence interval 33, 40–42

CPT Lab Computational Pharmacology & Toxicology Lab 2, 3

DILI drug-induced liver injury 14, 18

ECE expected calibration error 32, 33, 38, 40–42

FFN feed forward neural network 29, 30

GAT graph attention network 18

GELU Gaussian error linear unit 27–29

GNN graph neural network 5, 16–18, 20, 45–47

GPU graphics processing unit 29

GTN graph transformer network 5, 9, 18–20, 23, 25, 49, 50

H-bond Hydrogen Bond 62

IATA Integrated Approaches to Testing and Assessment 7

ICH International Council for Harmonisation 6, 7

LR learning rate 12, 27, 29

MACCS Molecular ACCess Systems 7, 8

MAE mean absolute error 46

MF molecular fingerprint 7–9, 14–18, 22, 35, 44–46, 48, 62

ML machine learning 13–15, 34

Morgan FP Extended-connectivity Fingerprint 4 (ECFP-4) 8

NIHS-J National Institute of Health Sciences, Japan 20, 21

NLP natural language processing 13, 24

NN neural network 5, 9, 10, 14–16, 44–46, 48

OECD Organisation for Economic Co-operation and Development 33

PCA principal components analysis 5, 22, 35, 36, 43, 44

QSAR quantitative structure activity relationship 5–8, 14, 18, 20, 33

REACH Registration, Evaluation, Authorisation and Restriction of Chemicals 7

SMILES Simplified Molecular-input Line-entry System 20, 22

SOTA state of the art 17–19, 46, 49, 50

SPD shortest path distance 24, 47, 48

UMAP Uniform Manifold Approximation and Projection 22, 23, 35, 36, 44

US FDA the United States Food and Drug Administration 7

WL test Weisfeiler-Lehman test 46–48

Algorithm 2 Centrality Encoding for Graphormer

Input: $G = (V, E)$, a graph with nodes V and edges E
Input: $X = \{x_1, x_2, ..., x_n\}$, node features for each node in V
Input: Z^-, Z^+, learnable embedding vectors for indegree and outdegree
Output: $H^{(0)}$, updated node features with centrality encoding
for each node $v_i \in V$ **do**
    Calculate $deg^-(v_i)$, the indegree of node v_i
    Calculate $deg^+(v_i)$, the outdegree of node v_i
    if the graph is undirected **then**
        Unify $deg^-(v_i)$ and $deg^+(v_i)$ into $deg(v_i)$
    end if
end for
for each node $v_i \in V$ **do**
    Assign centrality embedding based on degree centrality:
    $Z_{deg^-}(v_i) = Z^- \cdot deg^-(v_i)$ ▷ Centrality embedding for indegree
    $Z_{deg^+}(v_i) = Z^+ \cdot deg^+(v_i)$ ▷ Centrality embedding for outdegree
    if the graph is undirected **then**
        Use $Z_{deg}(v_i) = Z \cdot deg(v_i)$ where Z is a unified embedding vector
    end if
end for
for each node $v_i \in V$ **do**
    $H_i^{(0)} = x_i + Z_{deg^-}(v_i) + Z_{deg^+}(v_i)$
    if the graph is undirected **then**
        $H_i^{(0)} = x_i + Z_{deg}(v_i)$
    end if
end for
Incorporate $H^{(0)}$ into the Transformer model's attention mechanism

.1 Algorithms

.1.1 Encodings

.1.2 Learning Rate Schedulers

Algorithm 3 Linear Learning Rate Scheduler

Input: Initial learning rate η_{start}, final learning rate η_{end}, total epochs N

Output: Adjusted learning rate $\eta(t)$ for each epoch t

for $t = 0$ to N **do**

    $\eta(t) \leftarrow \eta_{\text{start}} + \left(\frac{t}{N}\right) \cdot (\eta_{\text{end}} - \eta_{\text{start}})$

    update model parameters using $\eta(t)$

end for

Algorithm 4 GreedyLR Scheduler

procedure GREEDYLR(optimizer, factor, patience, cooldown, warmup, minLR, maxLR, smooth, window, res
    Initialize: $lr \leftarrow$ initLR, bestLoss $\leftarrow \infty$,
        warmupCounter $\leftarrow 0$, cooldownCounter $\leftarrow 0$,
        numGoodEpochs $\leftarrow 0$, numBadEpochs $\leftarrow 0$
    trainLoss, valLoss $\leftarrow$ function to train model for one epoch
    if smooth is **True then**
        valLoss $\leftarrow$ streaming average of valLoss in window
    end if
    if valLoss < bestLoss **then**
        bestLoss $\leftarrow$ valLoss
        numGoodEpochs $\leftarrow$ numGoodEpochs + 1
        numBadEpochs $\leftarrow 0$
    else
        numGoodEpochs $\leftarrow 0$
        numBadEpochs $\leftarrow$ numBadEpochs + 1
    end if
    if cooldownCounter < cooldown **then**
        cooldownCounter $\leftarrow$ cooldownCounter + 1
        numGoodEpochs $\leftarrow 0$
    else
        cooldownCounter $\leftarrow 0$
    end if
    if warmupCounter < warmup **then**
        warmupCounter $\leftarrow$ warmupCounter + 1
        numBadEpochs $\leftarrow 0$
    else
        warmupCounter $\leftarrow 0$
    end if
    if numGoodEpochs > patience **then**
        $lr \leftarrow \frac{lr}{\text{factor}}$
    end if
    if numBadEpochs > patience **then**
        $lr \leftarrow lr \times$ factor
    end if
    if reset > 0 **and** epochs > reset **then**
        $lr \leftarrow$ initLR