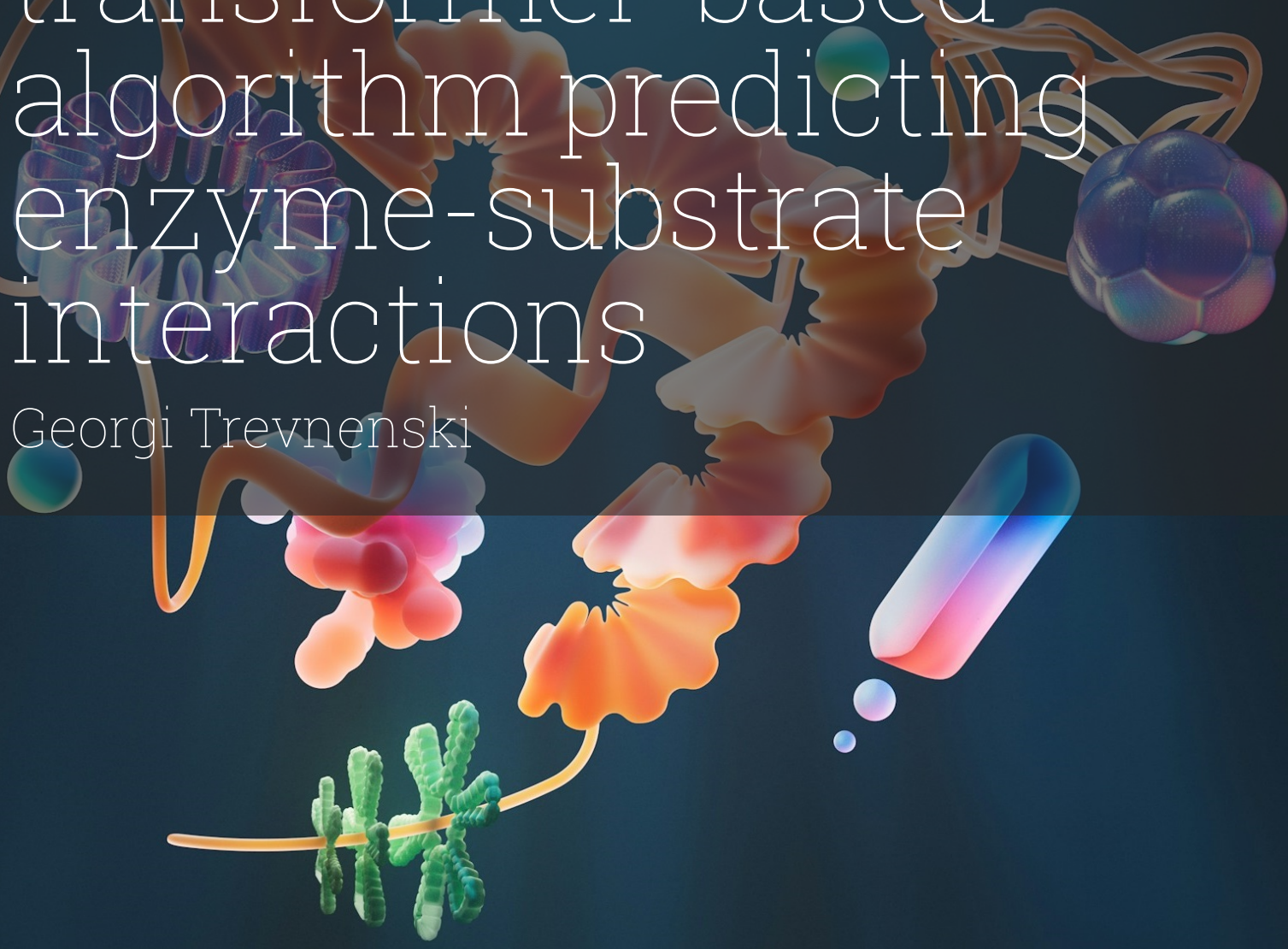
Modality fusion strategies in a transformer-based algorithm predicting enzyme-substrate interactions

Georgi Trevnenski



Delft University of Technology

TU Delft

Modality fusion strategies in a transformer-based algorithm predicting enzyme-substrate interactions

by

Georgi Trevnenski

Supervisor: M. Reinders
Daily supervisor: J. Weber
Daily co-supervisor: L. Di Fruscia
Project Duration: Feb, 2024 - Nov, 2024
Faculty: Faculty of Electrical Engineering, Mathematics & Computer Science, Delft

Cover: Image by DeepMind available on Unsplash.com
Style: TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

Accurately predicting enzyme-substrate interactions is critical for applications in drug discovery, biocatalysis and protein engineering. Building upon the ProSmith algorithm, a machine learning framework with a multimodal transformer for protein-small molecule interaction prediction, this study introduces protein 3D structural data as an additional modality. To integrate this data, we explore additive and multiplicative modality fusion strategies without requiring retraining the original transformer from scratch. Our experiments demonstrate that while the incorporation of structural data does not offer improved performance in random splits, it has the potential to surpass ProSmith in challenging data splits involving unseen small molecules. Notably, the model shows better generalization for underrepresented substrates.

Contents

I. Introduction

Predicting the interactions between proteins and small molecules is a persisting challenge in the field of bioinformatics. Recent progress in solving it through machine learning methodologies has been instrumental in drug discovery and protein engineering. A survey by Bagherian et al. [2] provides a broad overview of the techniques utilized for this task, while Zhang et al. [21] point out the decades of work on each drug and the billions of dollars spent, challenges such algorithms could address. A subset of the protein-small molecule interactions, the enzyme-substrate binding, has received comparatively less attention but is applied in biocatalysis for the production of numerous chemicals (Wu et al., 2021 [20]). Additionally, a number of commercially available products already depend on synthetic biology, where such algorithms are essential (Voigt, 2020 [17]).

According to Kroll et al. (2023) [10], previous models lose valuable information when reducing the protein to a single vector representation and only after incorporating information about the small molecule. They suggest a transformer-based algorithm that processes the two input modalities simultaneously in a way that facilitates learning interactions between individual amino acids and small molecule segments.

As a base for our research serves the ProSmith (PROtein-Small Molecule InTeraction Holistic) algorithm, developed by Kroll et al. [10]. We explore strategies for incorporating an extra modality to the input, namely protein 3D structure data. It is well-established in molecular biology that information on the 3D structure of biological macromolecules [...] is essential for the mechanistic understanding of their function" (Carugo et al., 2023[4]). Our study aims to improve the generalizability and robustness of ProSmith by enriching the input data for the algorithm. We analyze modality fusion methods that do not require retraining of the transformer mode and our model shows the potential to outperform ProSmith when tested against substrates, rarely occurring in the training set.

II. Background

In this section, we are briefly introducing the inner workings of ProSmith and point out its shortcomings we are about to address with our research. Additionally, we discuss our choice of modality fusion methods.

ProSmith processes protein-small molecule interaction predictions by integrating protein and small molecule data into a multimodal Transformer network. The modalities being used are a string of characters representing amino acids and a small molecule representation in SMILES [18] notation. ProSmith first takes a protein sequence, tokenized at the amino acid level and converts these tokens into representation vectors (each vector corresponding to a single amino acid) using the pretrained ESM-1b model [12]. Similarly, the small molecule, represented by a SMILES string, is tokenized and mapped to vectors via the ChemBERTa2 model [1]. The protein and SMILES embeddings are concatenated into a single input sequence, with a classification token (cls) at the beginning and a separation token (sep) between the two modalities. This combined input is processed by the multimodal Transformer, which updates the embeddings through its attention mechanism, outputting an updated cls token that integrates information from both the protein and the small molecule. Finally, gradient boosting models are trained on different combinations of three elements: the updated cls token, the mean of the embeddings from the ESM-1b model and the mean of the ChemBERTa2 embedding vectors. The final interaction prediction is generated by taking a weighted average of the outputs from these models. The output of the model is in the form of a value of a biochemical constant describing the interaction or a binary prediction of whether the protein and small molecule would bind.

A significant drop in the performance of the ProSmith algorithm is reported when tested against data samples containing a substrate and/or a protein not seen during the training phase. Furthermore, the results on the same prediction task show that performance declines significantly, with error rates increasing approximately threefold when using a dataset roughly three times larger. Although these numbers are reported only for a regression task (refer to Tables 1 and 4 in Kroll et al. [10]), we assume that a similar trend could be seen in binary classification. We believe that both of these limitations could be addressed by providing structural information about the proteins. In this research, we focus on the binary classification task as the dataset provided was easier to work with.

A straightforward way to add an extra modality would be to produce numerical representations for the 3D structure of the protein and append them to the rest of the input to the multimodal transformer. However, we assume that this approach would change significantly the input space of the transformer and would only work if we retrain the model from scratch. As this is a very time- and resource-intensive task we aim to use a ProSmith

model pretrained on the BindingDB [13] dataset provided by Kroll et al. [10].

Indeed, the state-of-the-art techniques for modality fusion are attention-based, training transformers on datasets with millions of data points containing both modalities. Models such as the Chameleon (Team, 2024 [15]) and the Generalist Agent (Reed et al., 2022 [14]) exemplify the power of transformers in modality fusion but both of these have required training on datasets comprising millions of samples.

Instead, we opted to fuse the amino acid sequence and the 3D structure embeddings before feeding them to the multimodal transformer. As described in Jayakumar (2020) [8], we use Equations (1) and (2) to approximate the interactions between the two modalities. A detailed explanation of our approach can be found in Section III Methods: E (Modality Fusion).

III. Methods

A. Research questions

Our study aims to answer what is the effect of adding an additional modality to a protein language model. Our research's sub-questions can be formulated as: (i) how the modality fusion methods implemented here compare to each other in different experiment setups, (ii) can adding protein 3D structure data to the input lead to an even better performance than the one documented in the paper by Kroll et al.[10], (iii) can this implementation generalize better to unseen data and (iv) can it lead to more robust models performing well in low-data scenarios. To answer these questions, a new benchmarking dataset was prepared, along with a collection of experiments to test the model in different settings. The new components added to this study compared to the research of Kroll et al. are the new dataset that includes graph data, the selection of a graph embedding algorithm and a modality fusion module to add to the model. Additionally, we conduct a brief ablation study with experiments introducing input perturbations and excluding components of the algorithm. Thus, we provide a more in-depth understanding of the inner workings of the algorithm.

B. Problem definition

Our algorithm receives two different representations of a protein and a single representation of a small molecule as input and outputs a binary prediction of whether this small molecule would interact with the protein. In this study, we test the use case of enzyme-substrate binding prediction. As showcased by Kroll et al., such an algorithm could also generalize to other interactions of proteins and small molecules such as drug-target affinity prediction.

C. Input representation

The input format consists of three modalities describing proteins and small molecules, which is visualized in Figure 1. To dive into the details of the input space we start by breaking down the protein data we use. It consists of strings of amino acid sequences and graphs representing the 3D structure of the proteins.

Amino acid sequence

An amino acid sequence is a string of characters, each representing a single amino acid. In total, 20 letters from the English alphabet are used to represent the 20 standard amino acids. The amino acid sequences are already available in the datasets published by Kroll et al [10].

Protein graphs

To complement the amino acid information, we opted to train the algorithm to make use of the 3D structure of the protein. For that purpose, we create a graph with nodes corresponding to amino acids for every protein we use. Thus, we can treat each amino acid as a separate token both in the sequence and the graph representation of the protein.

The Python library we made use of is called Graphein [7]. It can both download 3D structure data from AlphaFold's database [9] with coordinates per residue for each protein and convert it into a graph with nodes representing residues and edges based on interactions between pairs of amino acids. The interactions we consider for creating edges are based on peptide bonds and distance. Distance is a well-established metric used in protein graphs and in this case, we use as distance threshold 10 angstroms (10^{-10} m), which is on the higher end of the spectrum with Vendruscolo et al.[16] using a cut-off value of 8.5 angstroms. However, a second parameter is key when creating the edges and this is the "long interaction threshold" which is set to five. This implies that no distance-based edge will be added between any two nodes unless the corresponding residues are at least five positions apart in the amino acid sequence. This ensures that the distance-based edges are formed only for residues that are close together due to the way the protein has folded in space and not due to their proximity in the amino acid sequence. Addition-

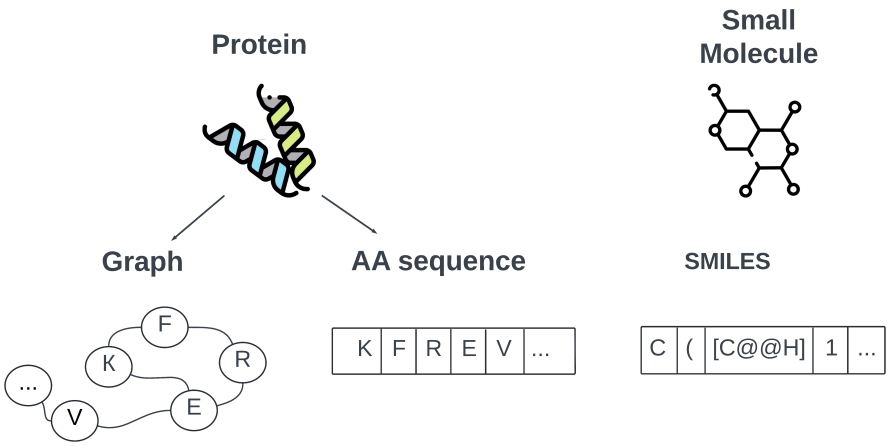


Figure 1: Illustration of the three input types used in the model: (1) protein structure graphs, (2) protein amino acid sequences, and (3) small molecule SMILES strings. The amino acid sequences provide linear information, while the graphs capture the 3D spatial relationships of amino acids. SMILES strings encode the chemical structure of the small molecule. These inputs are combined to predict interactions between the protein and the small molecule.

ally, the peptide-bond edges ensure that we keep the information about the sequence and that the graphs are connected.

SMILES strings

Small molecule data are represented through SMILES (Simplified Molecular Input Line Entry System) strings. SMILES notation is a way to represent chemical structures using short ASCII strings. Each atom is represented by its atomic symbol, bonds are implied or represented by specific characters (e.g., - for single, = for double), and rings are indicated by numbers. Branching is shown with parentheses. It provides a compact format for describing molecules which is easy to work with for an algorithm. This SMILES string is tokenized using the Chemberta2 model of Ahmad et al. [1].

To sum up, the data used as input for each protein-molecule pair consists of an amino acid sequence, a graph depicting the 3D structure of the protein and a SMILES notation describing the structure of the small molecule.

D. Dataset

We used the dataset created by Kroll et al. in a study on enzyme-substrate interaction prediction [11], in which they developed an algorithm they referred to as ESP (Enzyme Substrate Prediction). Together with the model, they composed the ESP dataset which includes positive enzyme-substrate pairs and negative pairs (with a 1:3 positive-to-negative ratio), resulting in almost 69 000 data points. The authors then further expanded the dataset by including data with phylogenetic evidence in addition to the data with experimental evidence so that the dataset gets to 850 000 data points. This dataset contains tuples of amino acid sequences (from around 144 000 unique proteins), SMILES sequences and binary output values indicating if the enzyme and the substrate bind together. It is important to note that the dataset the model was pretrained on has no overlap with ESP.

Structural information about proteins can be sourced from experimental data or from a computer model - the current state-of-the-art model being AlphaFold [9]. According to documentation from the Protein Data Bank (PDB) [3], both options have their downsides. Experimental structures can have limitations, such as mismatches with experimental data caused by errors in model construction, or regions that lack data due to disorder or movement. Additionally, distortions in atomic geometry, including deviations in bond lengths or bond angles may arise from model-building or refinement errors. Similarly, computed structural models (CSMs) also have limitations, including regions of low confidence due to data constraints. Even though experimental data is preferable in some cases we opted for only AlphaFold-modelled structures due to the ease of use of Python libraries providing interface to the model.

By using the Graphein library, we went through the UniProt IDs associated with the proteins in the ESP dataset and tried to download their 3D structures, which was successful in only a fifth of the cases forming a database with 30 248 unique proteins and 185 824 data points in total. Table 1 summarizes the important parameters of the aforementioned datasets. From now on we will refer to the 'expanded ESP' dataset as 'ESP', since we do not

Table 1: Comparison of the datasets used in the study. The two versions of ESP mentioned are provided by Kroll et al. [10]

Dataset	Has phylogenetic data	Has 3D structure data	#total points	#unique proteins
Original ESP	No	No	69,000	12,020
Expanded ESP	Yes	No	850,000	144,000
Our dataset	Yes	Yes	185,824	30,248

use the original one in the trainings.

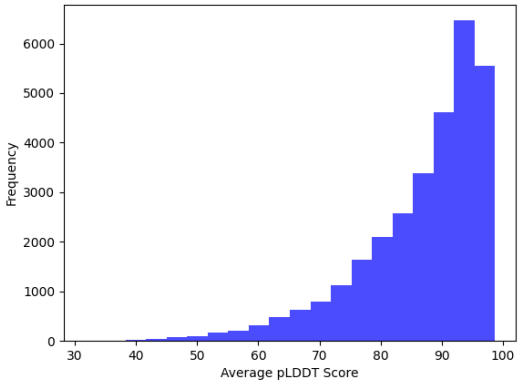


Figure 2: Distribution of average pLDDT (predicted Local Distance Difference Test) scores generated by AlphaFold for protein structures in the dataset. The pLDDT score ranges from 0 to 100, where higher values indicate greater confidence in the predicted structure. The scores are provided per amino acid so we average them out for each protein. A score above 90 suggests a highly reliable structure, while scores below 50 indicate low confidence.

It is not clear why the Python library cannot find all proteins from AlphaFold but we consider the ones we found enough for conducting our envisioned experiments. A larger dataset would make the experimentation within the scope of this master thesis more challenging given our computational constraints. Furthermore, we ensured that the downloaded protein 3D structures come from high-confidence AlphaFold predictions. The metric provided by AlphaFold for that is the predicted local distance difference test (pLDDT) which is a score from 0 to 100 with higher scores indicating higher confidence in the prediction. It is given on a per-residue basis so we averaged those out to obtain a score for each protein. A distribution of the results is shown in Figure 2. Based on the AlphaFold documentation, predictions with scores lower than 50 should be considered unreliable. We have only about 200 such entries with scores in the range of 40 to 50, which is less than 1% of all proteins we use. We chose to keep those as removing them would have required generating embedding files from scratch which is time-consuming. As illustrated in Figure 2, the largest partition of predicted protein structures has pLDDT scores of above 90%.

E. Model architecture

The architecture of the model can be divided into two phases. The first phase is illustrated in Figure 3.

Generating embeddings

The first stage is creating embedding vectors from the input tokens. The amino acid tokens and the SMILES tokens are converted to numerical encodings by the models ESM-1b and ChemBERTa2 respectively. Both are transformer-based pretrained models commonly used to create embeddings for protein sequences and SMILES strings. These models' weights are frozen during the training. We decided to use the same encoding models as in the research by Kroll et al. because we initialize the multimodal transformer with the weights provided by them. Thus, we hypothesized that changing the embedding layers could make it more difficult for the multimodal transformer to learn the new embeddings space.

The added graph tokens are encoded using the Node2Vec algorithm [6]. The mechanism by which it operates involves generating context for each

node in the graph through random walks. Context is defined as a set of nodes that are reached by random walks from the respective node in a set number of steps. The embeddings are generated by maximizing the probability that each context is generated from the particular node. Similarity is calculated through the dot product of the embedding vectors, based on the concept that nodes should resemble other nodes in their context. Other commonly used graph encoding techniques are deep learning algorithms such as Graph Convolutional Networks (GCNs) and graph factorization methods. However, GCNs are useful mostly when the algorithm needs to generalize to unseen nodes which is not the case here and factorization techniques are not performing well with large graphs. Node2Vec, on the other hand, is scalable and unlike deep learning techniques, it does not require pretraining and is not prone to overfitting.

The created embedding vectors have dimensions 1280 for the graph embeddings and 600 for the amino acid ones, as can be seen in Figure 3. Therefore, for each amino acid we have a vector encoding it as part of the protein sequence and a vector encoding its position in the protein graph (illustrated in red and yellow in Figure 3).

Modality fusion

We proceed by training a fusion module to merge the two vectors for each amino acid, representing the graph and the amino acid sequence embeddings, respectively. The reasoning behind this is that we want to keep the format of the input to the multimodal transformer as similar as possible. That way we hope to enrich the information of the vectors corresponding to each amino acid but not change completely the embedding space the multimodal transformer was pretrained to get its input from.

As formalized by Jayakumar et al. [8], in the general case for input modalities x and z, given $\mathbf{x} \in \mathbb{R}^n$ and $\mathbf{z} \in \mathbb{R}^m$, the aim is to model an unknown function $f_{\text{target}}(\mathbf{x}, \mathbf{z}) \in \mathbb{R}^k$. In our case, $n = m = k$. For modalities A and B with embedding vectors x_A and x_B, equation (1) shows a first-order polynomial describing an additive interaction or a weighted sum. Through training we obtain one weight vector per modality, we multiply the embedding vector by the corresponding weights and sum them. By adding a third term to the equation as it is in (2) we end up with a second-order polynomial or what we call additive and multiplicative fusion (it includes the terms from (1) and adds a multiplication term). Here we take into account the element-wise multiplication of the two embedding vectors. As described by Wörtwein et al. [19], one can add an arbitrary number of terms to such equation (e.g. $w_4(x_A^2 \times x_B^2)$) and higher order functions have higher expressive power to estimate the interaction between the two modalities. However, we run into the risk of overfitting. Thus, the order of the equation should be matched to the size of the training set and the training time. We use both equations (1) and (2) in separate experiments and compare the results with regard to the metrics discussed in the introduction of the results section. For simplicity, we refer to the model implementing equation (1) as **additive fusion**, while the model implementing equation (2) we refer to as **multiplicative fusion**. However, equation (2) includes all terms of equation (1) and could also be named "additive and multiplicative fusion". See Table 2 for an overview of the performance of these model variations.

$$z = w_1 \odot x_A + w_2 \odot x_B \tag{1}$$

$$z = w_1 \odot x_A + w_2 \odot x_B + w_3 \odot (x_A \odot x_B) \tag{2}$$

Moreover, in order to account for the different magnitudes in the embedding vectors coming from ESM-1b and Node2Vec, we devised experiments with two additional modifications to the algorithm. First, we test a model that normalizes the embedding vectors from both protein modalities based on the average size of the embedding vectors coming from each modality. Second, we opted for initializing the weight vector corresponding to the Node2Vec embedding vectors in equation (1) from a distribution with a mean 6.5 times larger than its counterpart weight, corresponding to how much larger the magnitude of the ESM-1b embedding vectors is on average compared to the graph embedding vectors. We refer to the first model as "additive/multiplicative fusion with **normalized embeddings**" and to the second modification as "additive/multiplicative fusion with **balancing weights**".

After the fusion model, the rest of the algorithm is structured as in Kroll et al. [10]. As seen in Figure 3, the vectors encoding protein data and the SMILES encoding vectors differ in size so they go through linear layers that project them to a common embedding space of dimension 768.

Multimodal transformer

The two sets of vectors are used as input to a Transformer encoder. To mark the end of one

modality and the beginning of the other, a separation token is used which is identical across input sequences (marked as 'sep' in the figure). Moreover, a classification token 'cls' is used to distill the valuable information from the two modalities and use this for classification further down the line. The separation token is initialized as a vector of zeros, while the classification token is initialized as a vector of ones.

The Transformer Network (level C in Figure 1) updates each input token through the attention mechanism, which allows the model to examine the entire input sequence and selectively focus on relevant tokens for updates. Once all input tokens have been updated for a specified number of steps, the classification token (cls) is extracted. This token is then fed into a fully connected neural network (noted as MLP in Figure 3) that is trained to predict the interaction between the small molecule and the protein. By training this part of the algorithm end-to-end, it learns to encode all essential information for the interaction prediction within the cls token.

After training the Transformer Network to predict protein-small molecule interactions, the cls token is extracted as a joint representation of the three modalities. However, given the limited size of the cls token, Kroll et al. [10] hypothesized that important general information might be lost. To address this, they incorporate direct information from the protein and small-molecule representation vectors. These representations are computed by taking the element-wise mean across the embedding vectors for each modality, creating an ESM-1b mean vector and a ChemBERTa2 mean vector. Additionally, we have one more modality, thus we calculate a third element-wise mean vector from the Node2Vec embeddings. As can been in Figure 4, those vectors are then processed by a gradient boosting algorithm.

Gradient boosting models

Previous research has shown that using learned representations from Transformer Networks as inputs for gradient boosting models further improves outcomes over using the model's predictions directly. Gradient boosting models use iterative decision trees that reduce the errors of the previous iterations. This study adopts a similar approach by using the cls token, ESM-1b, Node2Vec and ChemBERTa2 mean vectors as inputs for a gradient boosting model - namely XGBoost [5]. This is what we refer to as the second phase of the training.

To enhance model performance, it is common to train multiple models and use an ensemble of their predictions, often resulting in more accurate and robust outcomes. Thus, Kroll et al. hypothesize that training multiple gradient-boosting models with different input representations would improve predictions. They train three models: (i) using only the cls token, (ii) using the concatenated ESM-1b and ChemBERTa2 mean vectors, and (iii) using all three input vectors. The final prediction is obtained by computing a weighted mean of these models' predictions. Our modification to this section of the algorithm is to add the mean vector of the Node2Vec embeddings to the other mean vectors in model (ii).

F. Training of the model

In this section, we discuss the details of the model's implementation and training.

Calculating the embeddings

As mentioned before, to encode the protein sequences and SMILES strings we use the same pretrained models as Kroll et al. [10]. Proteins are encoded by the ESM-1b model which is a Transformer Network with 33 layers, trained on about 27 million protein sequences. While the SMILES strings are processed by ChemBERTa2 - a 3-layer model trained on around 77 million different SMILES strings.

Next, in the fusion module the weights in equations (1) and (2) are initialized equal, drawn from a normal distribution with a mean of 1 and a standard deviation of 0.02.

Furthermore, it's worth mentioning that the maximum length for protein sequences is capped at 1024 tokens, and for SMILES strings, at 256 tokens. If a sequence exceeds these limits, only the first 1024 amino acids or the first 256 SMILES tokens are retained and passed through the embedding modules. In our dataset, this leads to 9246 protein sequences (incl. duplicates) from the training set of a random split being trimmed. However, no SMILES strings are exceeding the respective limit.

Input to the multimodal transformer

Before entering the multimodal Transformer Network, the embedding vectors pass through single-layer fully connected neural networks with a ReLU activation function, which maps them to a common dimensional space. The input sequence in the multimodal Transformer Network then follows the structure shown in Figure 3: it begins with the 'cls' classification token, followed by tokens representing the protein, a separation token, and finally, the

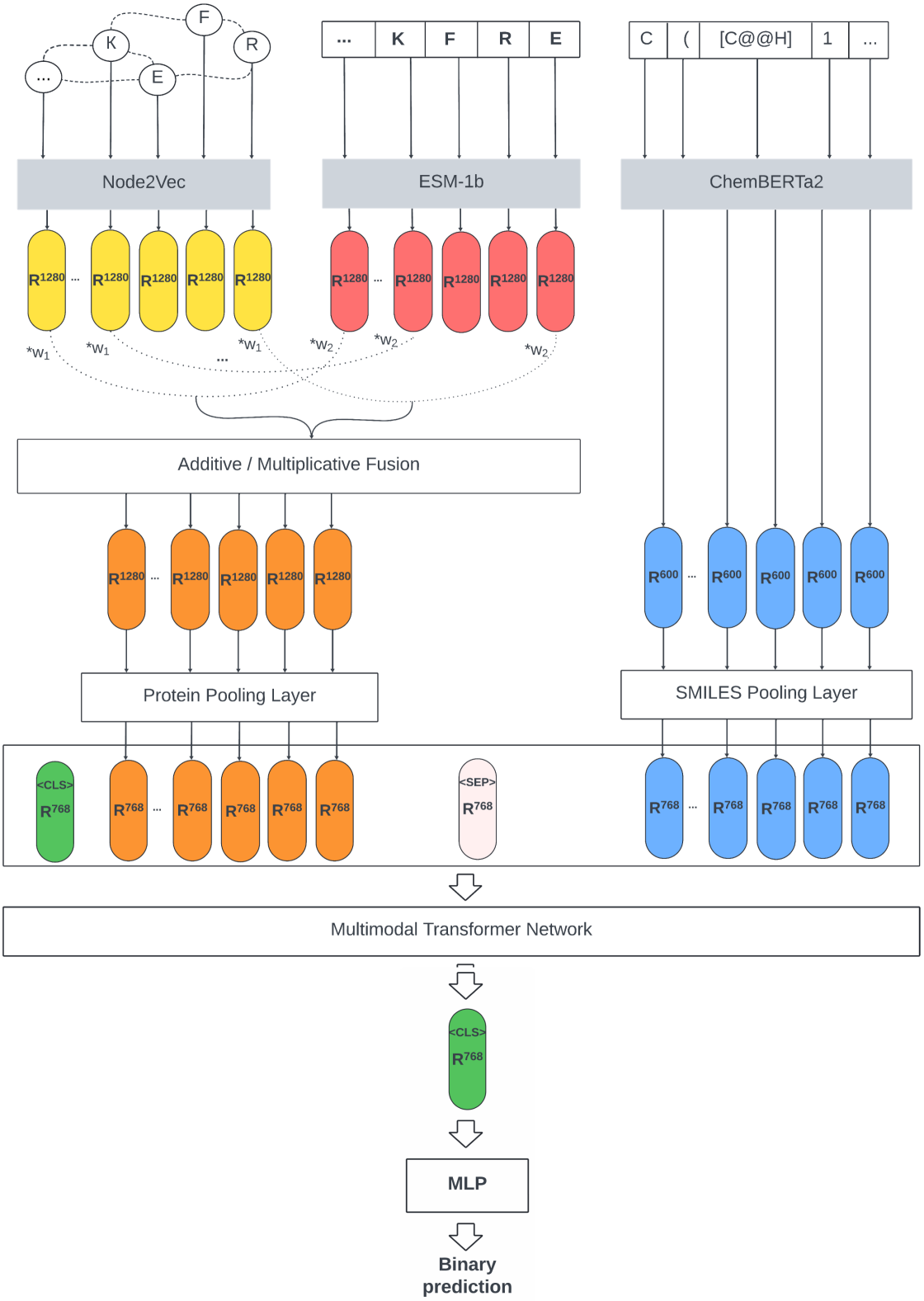


Figure 3: Transformer model architecture. The model accepts three different types of inputs: (1) amino acid sequences of proteins, (2) protein structure graphs, and (3) SMILES strings of small molecules. The protein sequence and SMILES tokens are encoded by the pretrained ESM-1b and ChemBERTa2 models, respectively, into embedding vectors. Simultaneously, the protein graph is encoded using the Node2Vec algorithm to capture structural information. A fusion module merges the protein sequence and graph embeddings for each amino acid. These embeddings are then projected into a common dimensional space. The fused protein and SMILES embeddings are concatenated and fed into a Transformer network, which processes the combined input through multiple layers of attention and produces an updated classification token ('cls'). This 'cls' token is used to make predictions about protein-small molecule interactions. This architecture is trained end-to-end except for the embedding modules, which are colored in grey.

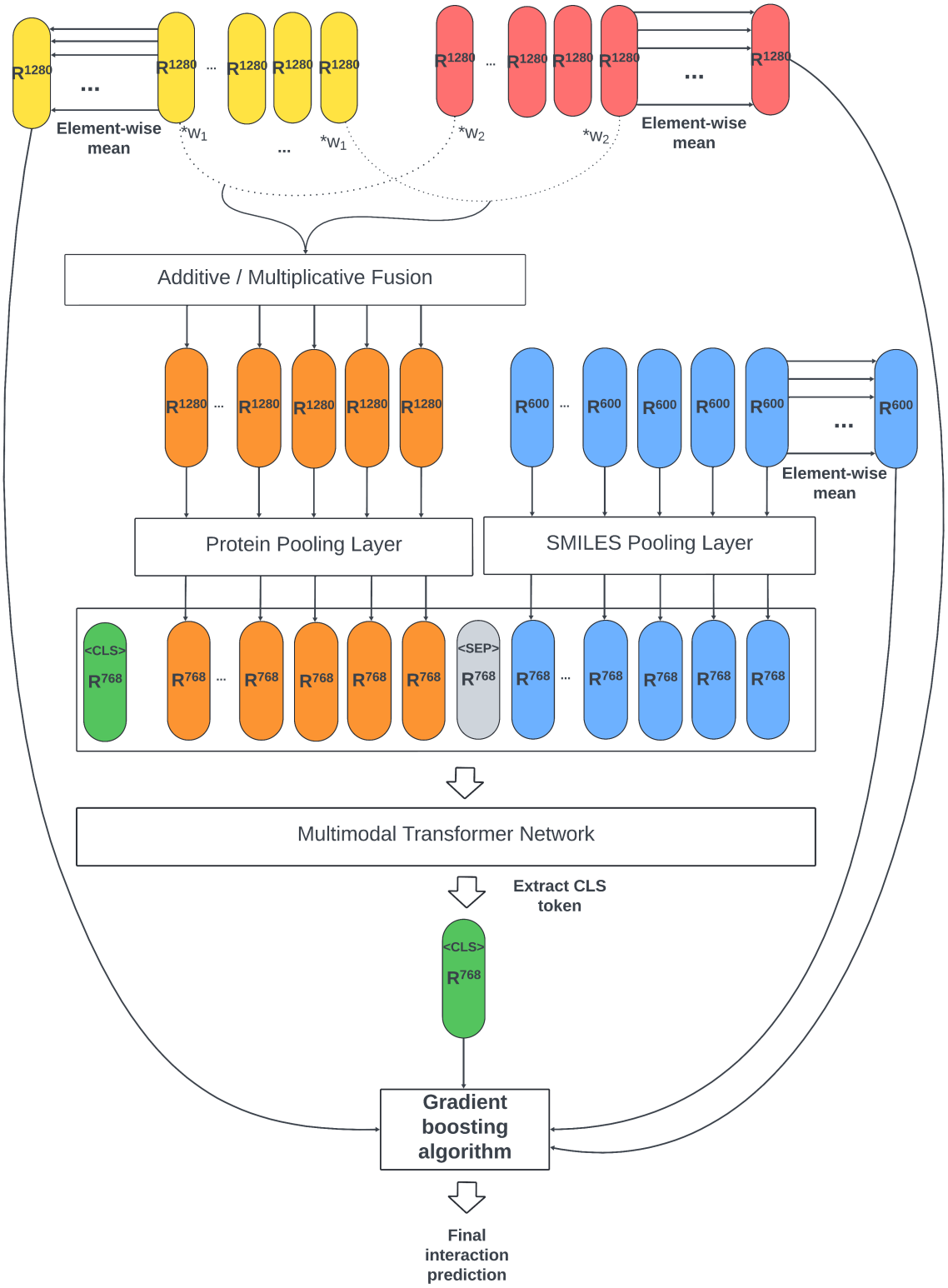


Figure 4: Gradient boosting model architecture. The updated 'cls' token, together with the element-wise means of the embedding vectors from each modality are used as inputs for multiple gradient boosting models. Each model is trained on different combinations of these input representations. The predictions from these models are then combined using a weighted average to generate the final interaction prediction. This ensemble approach helps improve model accuracy and robustness by leveraging complementary information from different input representations.

tokens corresponding to the SMILES string. The 'cls' token is encoded as a vector of all ones, while the separation token is encoded as a vector of all zeros.

The multimodal transformer in Figure 3 is a BERT (Bidirectional Encoder Representations from Transformers) model. It was configured with 6 layers, each containing 6 attention heads. It employed the GELU activation function, a smoother alternative to ReLU. After processing the input tokens through the six layers, a 768-dimensional representation of the classification token is extracted and passed into a fully connected neural network with a hidden layer of size 32 using the ReLU activation function (denoted as MLP in Figure 3). The output layer consists of a single node with a sigmoid activation function.

Training parameters

The whole pipeline showcased in Figure 3, excluding the Node2Vec, ESM-1b and ChemBERTa2 models is trained together end-to-end. The learning rate is established at 10^{-5}. Binary cross-entropy loss is applied as most commonly used for binary classification tasks. The training process was conducted for 20 epochs, with model parameters saved at the end of each epoch. Although Kroll et al.'s original implementation was trained for 100 epochs on the ESP dataset, we observed that our training converged around the 20th epoch. Kroll et al.'s paper does not provide plots illustrating the training progress, so a direct comparison is not possible. However, we expected earlier convergence in our training process because our dataset is approximately five times smaller than the original ESP dataset. The batch size was kept at 24 data points as it is in the work by Kroll et al.

Furthermore, the dataset is split into 80% training, 10% validation and 10% test data. During the first stage of the algorithm (illustrated in Figure 3), the training and validation sets are used to train the model for generating the classification token. In the second stage of the algorithm (Figure 4), the test set is also used to make an unbiased estimate of the error.

Batch processing

Storing all protein sequence tokens and SMILES string tokens simultaneously during training consumes excessive RAM for large datasets. To address this, Kroll et al divided the protein sequences and SMILES sets into smaller subsets of 2,000 each. We stick to that approach and create sets of graphs that correspond 1:1 with the protein sequence sets. During training, only one subset of tokens from each modality is loaded into RAM at a time, and the training iterates over all possible combinations of these subsets. The subsets are loaded in the same order on every epoch, however, the data points are shuffled within each subset.

Pre-training of the transformer network

We load a transformer model trained by Kroll et al. The authors used the Ligand-Target-Affinity dataset from BindingDB46, focusing on drug-target pairs with experimentally measured IC50 values. The dataset they curated has 1 039 565 entries and they trained the algorithm for 100 epochs with a batch size of 192 and a learning rate of 1.5×10^{-5}.

Training the gradient boosting models

The research of Kroll et al. [10], includes optimization of all hyperparameters for the gradient boosting models: learning rate, maximum tree depth, lambda and alpha coefficients for regularization, maximum delta step, minimum child weight, and number of training epochs. We used the same implementation based on the Python package xgboost[5] and the same hyperparameters as found optimal by the original paper, except the number of training epochs which we reduced in half to 250 due to time limitations.

IV. Results and discussion

We evaluate the performance of our algorithm using the metrics utilized by Kroll et al. [10] in order to compare to their implementation, as well as introduce new experimental setups to answer all of our research questions. First, we will briefly discuss the metrics used.

The metrics used for the final classifier after the gradient boosting algorithm are accuracy, MCC (Matthew's Correlation Coefficient) and ROC-AUC (Receiver Operator Curve - Area Under the Curve). Accuracy measures the proportion of correct predictions out of the total predictions. MCC evaluates the correlation between the predicted and actual binary classifications, providing a value between -1 and +1, with values close to +1 indicating a strong positive correlation, values near 0 indicating no correlation, and values close to -1 indicating a strong negative correlation. ROC-AUC measures the extent to which the classifier is able to separate the two classes, with an area under the curve closer to 1.0 indicating better discrimination between the positive and negative classes. For all three of those metrics, a higher value is desirable.

In terms of the experiments, we opt out to test the algorithm in different data split and data size scenarios. As discussed in Kroll et al's paper, the algorithm performance drops significantly when tested against unseen proteins or molecules. They suggest four splits and we follow the same structure. First is a random data split which is an arbitrary division into 80% - 10% - 10% train, validation and test sets. We then define three additional scenarios: cold protein, cold SMILES, and cold protein and SMILES. In these cases, 'cold' indicates that the test set consists of interactions involving substrates, enzymes, or both that are deliberately held out from the training and validation sets. For example, in the cold protein split, the test set includes only proteins that were unseen by the algorithm during training and validation.

The next sections will address each one of our research questions, in the order they have been presented in section III Methods A. After, we have added sections on error analysis and ablation study.

A. Performance comparison of the modality fusion strategies

During the first training phase (no gradient boosting) with our strategy, we observe comparable behaviour to the Prosmith architecture during the training process, as can be observed from Figure 5, a trend consistent across all fusion strategies we tested. The data split is irrelevant in this phase of the training as it only affects the test set and here only the accuracy of the validation set is plotted.

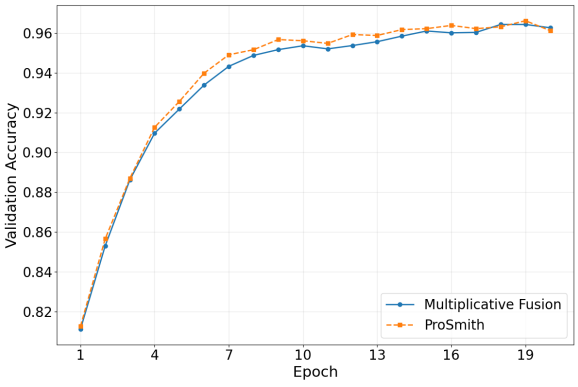


Figure 5: Validation set accuracy for the multiplicative fusion strategy and ProSmith during the first phase of training on the random split, without embedding normalization or weights adjustments.

Our results for cold data splits are reported in Table 2. We chose to test the performance of our model extensively on the cold SMILES and cold protein & SMILES data splits as Kroll et al. [10] have reported performance degradation for these specific splits. Furthermore, results for the cold protein split are in line with the random split performance, both for the Prosmith algorithm as well as our implementation, suggesting that new proteins do not pose a serious challenge to these models. Further details on that in subsection B, "Comparison of the new implementation against the original algorithm".

From the results reported in Table 2, we observe that the accuracy does not vary significantly among the different tested setups. However, the ROC-AUC values are generally higher for the cold SMILES split. Additionally, all experiments showcase slightly negative MCC values, the causes for which we will discuss in the following subsection C.

Based on these results, the multiplicative fusion model with normalized embeddings performs slightly better than the rest. However, we highlight the tiny margins, sometimes only 0.1%, with which it stands out compared to the second-most accurate model. To have more conclusive results, averaging these values among multiple runs of the experiments is crucial. Due to limited time and computing resources, this study provides results from single runs of these experiments only.

B. Comparison of the new implementation against the original algorithm

To have a new baseline to compare our implementation with, we opted to first run the original algorithm by Kroll et al. [10] on our newly composed dataset. Naturally, we don't make use of the graph data for the ProSmith algorithm, as it only works on protein amino acid sequences and SMILES strings. Therefore, the only difference is in the size and diversity of the datasets and these are showcased in Table 1. The ESP dataset is the one thoroughly discussed in Kroll et al.'s paper and the one to which we compare our results.

The only direct comparison to be made with the results in Kroll et al.'s paper can be seen in Table 3. First, we compare their algorithm on the random split of the ESP dataset and on our dataset. The results on our dataset are slightly higher in every metric. Additionally, we select our best-performing model from Table 2 (multiplicative fusion model with normalized embeddings) and demonstrate that its performance on the random split closely aligns with the original model's results without surpassing them.

Table 2: Comparison of all fusion techniques with both normalized embedding vectors and balancing weights. All setups are tested against the ”cold SMILES” and ”cold protein and SMILES” data splits of our dataset. Additive fusion is abbreviated to A.F. and multiplicative fusion to M.F.

Setup	Data split	Accuracy	ROC-AUC	MCC
A.F.	cold SMILES	72.6%	0.400	-0.055
A.F.	cold protein & SMILES	71.8%	0.258	-0.087
M.F.	cold SMILES	72.3%	0.420	-0.049
M.F.	cold protein & SMILES	72.1%	0.230	-0.084
A.F. with balancing weights	cold SMILES	72.6%	0.404	-0.062
A.F. with balancing weights	cold protein & SMILES	73.9%	0.244	-0.047
A.F. with norm. embeddings	cold SMILES	73.0%	0.395	-0.044
A.F. with norm. embeddings	cold protein & SMILES	73.2%	0.336	-0.063
M.F. with norm. embeddings	cold SMILES	73.0%	0.413	-0.050
M.F. with norm. embeddings	cold protein & SMILES	74.0%	0.273	-0.042

Table 3: Results obtained on a **random** data split. Performance comparison of ProSmith on the original dataset (results documented in Kroll et al. [10]) and on our dataset, as well as our multiplicative fusion model with normalized embeddings.

Model	Dataset	Accuracy	ROC-AUC	MCC
ProSmith	ESP	**94.2%**	**0.85**	**0.972**
ProSmith	Our Dataset	**97.6%**	**0.993**	**0.939**
M.F. with norm. embeddings	Our Dataset	**97.3%**	**0.992**	**0.931**

The close-to-perfect results can be explained by the already very high performance of the model in random data split scenarios and with the smaller dataset the algorithm can more easily learn. Furthermore, we are introducing a selection bias in the dataset by having only proteins for which the Graphein library could download 3D structures. This could potentially mean that these are commonly occurring or more well-known proteins. However, we cannot confirm this theory.

Likewise, Table 4 juxtaposes the multiplicative fusion model with normalized embeddings with the ProSmith model on the cold protein split. Interestingly enough, the results are very close to the random split, indicating the algorithm generalizes very well to unseen proteins as long as the SMILES strings in the data points have appeared in the training. In this data split, ProSmith achieves an accuracy of 97.0%, while our algorithm trails slightly with an accuracy that is 0.1% lower.

Table 4: Results obtained on a **cold protein** data split of our dataset. Performance comparison between ProSmith and our multiplicative fusion model with normalized embeddings.

Model	Accuracy	ROC-AUC	MCC
ProSmith	**97.0%**	**0.993**	**0.922**
M.F. with norm. embeddings	**96.9%**	**0.992**	**0.921**

Coherently with what has been done by Kroll et al. as well, we further investigate the contribution of the different inputs of the gradient-boosting model to the final performance. In Figure 6 we see the contributions of each of the models for the ProSmith algorithm as illustrated in Figure 4. We can examine how these weights change for each data split. Overall, we can confirm that both the mean embedding vectors and the cls-token are taken into account in the final decision in all data splits. Other than that, no clear correlation can be found between the weighting of the models and the accuracy of the predictions.

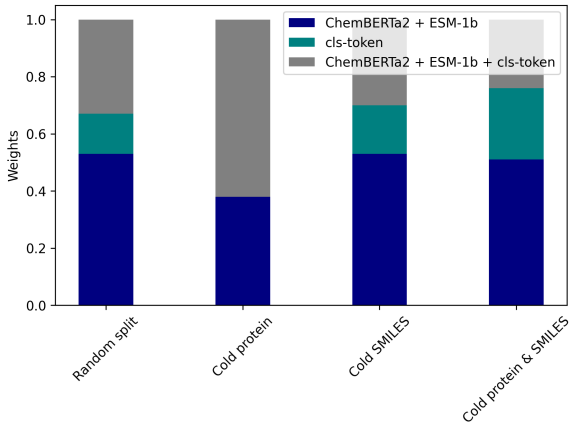


Figure 6: Model contributions to the optimal classifier in the gradient boosting phase as part of ProSmith. Bar plots illustrate the weights assigned to predictions from three gradient boosting models: cls token only, ESM-1b and ChemBERTa2 vectors, and all three input vectors combined.

Similarly, Figure 7 illustrates the weight distribution among the models in the ensemble for the multiplicative fusion with normalized embeddings algorithm. One noticeable difference with Figure 6 is the larger weight on the model relying only on the mean embedding vectors. This likely means that the multimodal transformer struggles with learn-

ing the new embedding space with the fused protein modalities. Thus, the model compensates by putting a larger weight on the mean embeddings instead of the classification token produced by the multimodal transformer.

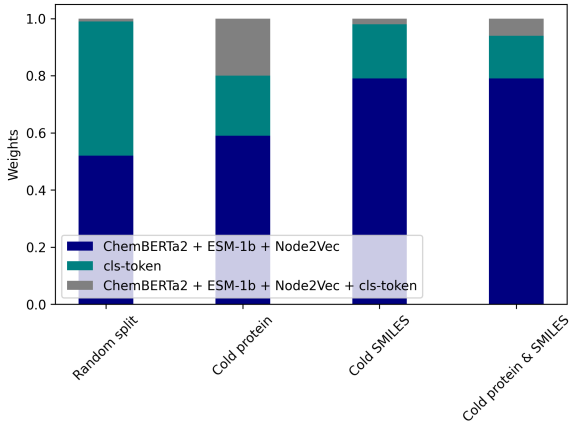


Figure 7: Model contributions to the optimal classifier in the gradient boosting phase as part of the multiplicative fusion model with normalized embeddings. Bar plots illustrate the weights assigned to predictions from three gradient boosting models: cls token only, ESM-1b and ChemBERTa2 vectors, and all three input vectors combined.

C. Can the new implementation generalize better to unseen data

Although Kroll et al. do not provide results from their study on other data splits for the ESP dataset, a comparison of the cold splits for a regression task can be found in their Supplementary Table 6 [10]. In their analysis, the coefficient of determination (R^2) drops significantly from 0.77 for a random split to 0.1 for a cold drug and target split (the dataset in question pertains to drug-target affinity). This sharp decline in performance ultimately means that their model is ineffective for this particular dataset and split.

Therefore, improved performance on such cold data splits was our main priority with this study. To compare side by side our most accurate model from Table 2 with the ProSmith algorithm, we have composed Table 5. Even if the difference is not striking, our implementation is performing slightly better on all three metrics and the accuracy in particular shows an increase of almost 1% in both splits.

However, despite the relatively high accuracy, the MCC values remain slightly below 0 across both data splits. The reason behind this is the imbalanced dataset we are working with. For example, the test set of the cold SMILES data split has 75.26% negative data points. The confusion matrix for the test set, presented in Table 6 showcases that the model outputs ”False” above 96% of the cases. Even though this is correct roughly 75% of the time, it still causes a large number of False Negatives (FN) and besides that there are more False Positives(FP) than True Positives (TP). For all these reasons, in the computation of the MCC (Equation 3) we observe that the product of the off-diagonal elements in the confusion matrix overcomes the diagonal elements and leads to a negative value.

Table 6: Confusion matrix of the multiplicative fusion model with normalized embeddings on the cold SMILES data split. Abbreviations: TP-True Positive, FP-False Positive, TN-True Negative, FN-False Negative

	Predicted Positive	**Predicted Negative**
Actual Positive	55	3447
Actual Negative	382	10273

$$\text{MCC} = \frac{(TP \times TN) - (FP \times FN)}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad (3)$$

After observing negative MCC values, we verify if our model performs any better than a random classifier with knowledge of the prior distribution. Figure 8 demonstrates the classification power of the multiplicative model with normalized embedding vectors on the data points involving the top ten most commonly occurring SMILES in the test set of the cold SMILES split. Over the bars in the plot, we see the performance of the ProSmith model and the random classifier marked with dotted lines. The random classifier achieves lower accuracy in all bins, however, as noted on the right it has the highest MCC score of the three algorithms. This trend might not hold for less common SMILES, however, the ones displayed in Figure 8 already account for around 61% of the test set.

To analyze further the generalizability of our model, Figure 9 displays how the MCC changes as a function of how many times a substrate has occurred in positive samples in the training set. Our model significantly outperforms ProSmith on this metric for rarely occurring substrates (up to 8 occurrences in positive samples). This constitutes only 4.3% of this test set and thus does not make a big difference towards the overall performance. However, we highlight the fact that even though this is not a cold data split, our model reaches very high MCC values for the underrepresented substrates and especially striking is the difference in the group of data points with substrates never occurring in pos-

Table 5: Performance comparison of ProSmith and the multiplicative fusion with normalized embeddings model on cold SMILES and cold protein & SMILES data splits.

Model	Data split	Accuracy	ROC-AUC	MCC
ProSmith	Cold SMILES	72.1%	0.411	-0.056
M.F. with norm. embeddings	Cold SMILES	73.0%	0.413	-0.050
ProSmith	Cold protein & SMILES	73.1%	0.251	-0.055
M.F. with norm. embeddings	Cold protein & SMILES	74.0%	0.273	-0.042

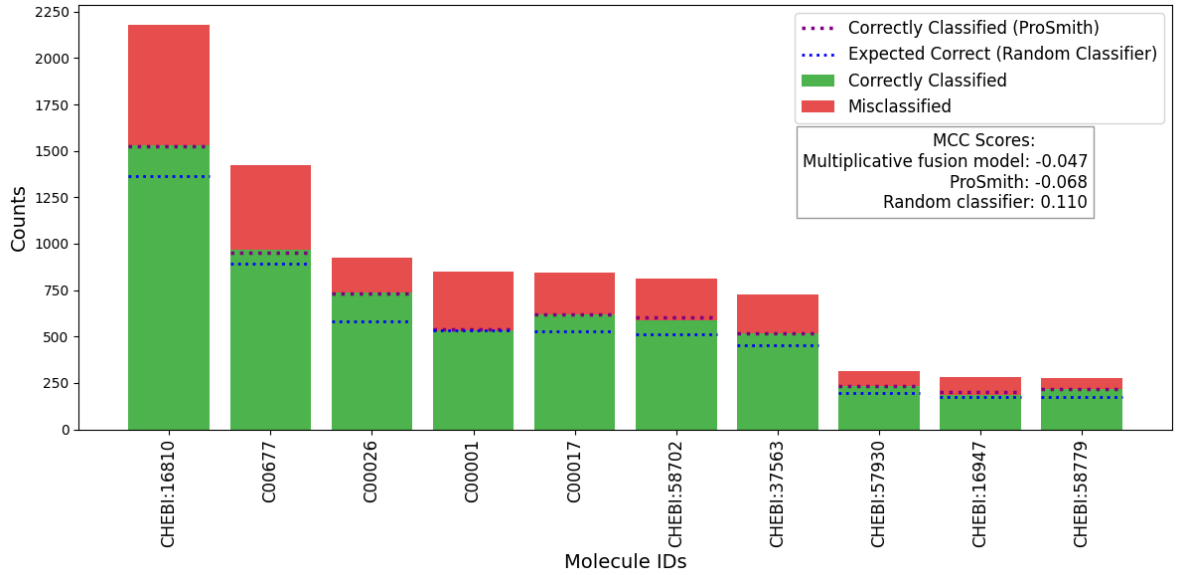


Figure 8: Performance evaluation of the multiplicative fusion with normalized embeddings model on the top 10 most common substrates in the cold SMILES split.The performance of the ProSmith model and a random classifier with prior distribution knowledge are also indicated for comparison.

itive training samples. Given how naturally imbalanced such datasets are, we consider this a very useful property.

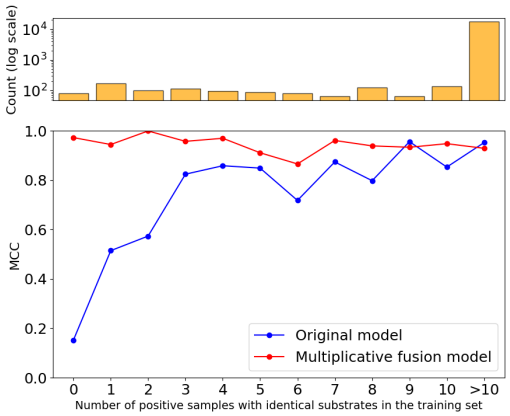


Figure 9: Our model consistently outperforms ProSmith in classifying data points with substrates that appear in the training set a limited number of times. MCC performance as a function of how many times a substrate occurred in a positive sample in the training set of the random split.

D. Performance in lower data scenarios

To answer our final research question of how our model will perform with limited training data, we conduct a set of experiments with 20% and 10% of the available data in our dataset. The ratios of training, validation and test sets remain the same at 80% - 10% - 10% respectively. Since we could not reach a significantly better performance than a random classifier on the cold SMILES and cold protein & SMILES, we concluded that reduced datasets are unlikely to yield meaningful results on these splits. Therefore, we only worked with a random and cold protein data split for these experiments. The results from the lower data setups can be seen in Table 7. Due to limited time, we only tested the multiplicative fusion model with normalized embeddings as the best-performing model from our previous experiments. The ProSmith model consistently outperforms our model across all metrics and scenarios. The differences in performance actually get larger as the data gets more scarce, meaning that ProSmith definitely is to be preferred in such scenarios. This can be explained by the larger number of parameters in our model due to the additional modality fusion module, which likely increases the risk of overfitting on smaller datasets. However, more experiments would be needed to make any strong statements on these setups. The smaller the data sets, the more unstable the training gets and the more variable results one obtains normally. Therefore, multiple runs need to be made in order to draw conclusions.

E. Graph and error analysis

Graphs are a crucial component in our implementation as they are the new modality we add to the algorithm's inputs. Table 8 provides common graph metrics for our dataset. On average, each protein graph contains approximately 500 nodes and 2,858 edges, with substantial variability observed in both metrics. Furthermore, the average density is low, around 0.03, suggesting that the graphs are generally sparse. Additionally, the graph diameters range widely from 5 to 618, with an average of 39, suggesting that some graphs are compact while others have much longer paths, indicating there are proteins not so tightly folded. Figures A.1 and A.2 in the appendix reveal the distributions in our dataset of the number of nodes and edges respectively. Overall, the dataset contains a diverse set of protein graph structures, with varying degrees of complexity and connectivity.

To investigate the misclassified proteins by our algorithm, we gathered the same statistics as in Table 8 for the unique proteins in the misclassified samples from the test set of the random split, when using the multiplicative fusion model with normalized embeddings. The statistics obtained are very close to the ones for all available proteins. None of the values deviates by more than a standard deviation from the respective mean found in Table 8. A full overview of the statistics on the misclassified proteins can be seen in Table A.1 of the Appendix.

F. Ablation study

To get a better grasp of the robustness of the model and its inner workings, we performed an ablation study. First, we performed modality fusion without our fusion module. Instead of fusing the Node2Vec and the ESM-1b embedding vectors first and then feeding the output to the multimodal transformer, we directly concatenated the Node2Vec embeddings right after the ESM-1b embeddings in the input sequence to the transformer. Thus, we rely on the attention mechanism of the transformer to perform the modality fusion. We opted to conduct this experiment only with the cold SMILES data split due to time constraints. In this run, we observed that the validation accuracy during the first phase of the algorithm (illustrated in Figure 3) reached 96.5% which approaches closely the results of the other models discussed previously. As the test set in the random data split has the same data distribution we can expect a performance of above 96% after the gradient boosting in this data split. However, the final accuracy of the model tested in a cold SMILES scenario is 72.7%, falling just 0.3% short of the best results displayed in Table 2. Although we lack enough evidence to make a strong argument about whether this setup performs better or worse than the one we proposed earlier, we can conclude that the multimodal transformer demonstrates remarkable robustness. Even such a major perturbation in the input sequence does not significantly impact its performance.

Next, we opted to test the significance of the gradient boosting section of the model (illustrated in Figure 4. By default, the models are run against the test set only after the gradient boosting algorithm. Therefore, we see a larger difference in the accuracies before and after the gradient boosting for the cold splits where the test set is the challenge. In this setup, we ran the algorithm on the test set of the cold smiles data split directly after the first phase of the training. The accuracy obtained is 71.5% with ROC-AUC standing at 0.5. This shows a 1.5% drop in accuracy compared to the best re-

Table 7: Performance comparison of ProSmith and the Multiplicative Fusion model with normalized embeddings in lower data scenarios. We consider partitions with 20% and 10% of all data points in our dataset.

Model	Dataset partition	Data Split	Accuracy	ROC-AUC	MCC
ProSmith	20%	random	90.0%	0.942	0.734
M.F. with norm. embeddings	20%	random	88.1%	0.923	0.678
ProSmith	20%	cold protein	89.8%	0.948	0.738
M.F. with norm. embeddings	20%	cold protein	89.3%	0.938	0.724
ProSmith	10%	random	87.2%	0.906	0.656
M.F. with norm. embeddings	10%	random	82.2%	0.844	0.507
ProSmith	10%	cold protein	83.8%	0.865	0.565
M.F. with norm. embeddings	10%	cold protein	83.8%	0.863	0.567

Table 8: Summary statistics for the graph properties of all proteins in our dataset.

	Num Nodes	Num Edges	Density	Average Degree	Diameter
Mean	500.11	2858.14	0.03	11.64	38.96
Std	323.22	1735.41	0.02	2.22	45.86
Min	23.00	35.00	0.00	2.00	5.00
Max	2696.00	17348.00	0.23	17.93	618.00

sult displayed in Table 2, even though the ROC-AUC value obtained is higher than previous results. We need to average the results across more runs of this experiment to make a strong conclusion on the contribution of the gradient boosting algorithm. Given our limited data, however, it appears that the gradient boosting step provides only marginal added value.

To conclude our ablation study, we experiment with replacing each of the protein modalities with noise. We generate noise vectors of the same dimensions and magnitude as the embedding vectors from the respective embedding modules - ESM-1b and Node2Vec. Then, we run the first phase of the algorithm three times with noise instead of amino acid sequence embeddings and the same with noise instead of graph embeddings. We have plotted the average accuracy and standard deviation across the three runs in Figures 11 and 11. The accuracy curve of the runs with noise instead of graph embeddings displays a clearer learning trend, reaching a value of 92% after 13 epochs. Meanwhile, the accuracy achieved with noisy amino acid sequence embeddings stands just above 68% after the same number of epochs. The results suggest that while graph information adds value, it is not as pivotal as the sequence-based embeddings. However, it could also be the case that the model needs significantly more epochs in order to converge when being trained with noise instead of ESM-1b embeddings. We decided to further verify whether our model (without any noise) during training learns to make use of both modalities or ignores one of them. To test this we loaded the trained multiplicative fusion model with normalized embeddings and calculated the cosine similarity of the fused vectors per amino acid with each of the corresponding embedding vectors before the fusion module. When averaged out over all amino acids of all data points in the dataset, the results showed that both have a value of around 0.3 cosine similarity with 1.0 meaning fully correlated vectors. Therefore, we can confirm that the model considers both modalities in the fusion step.

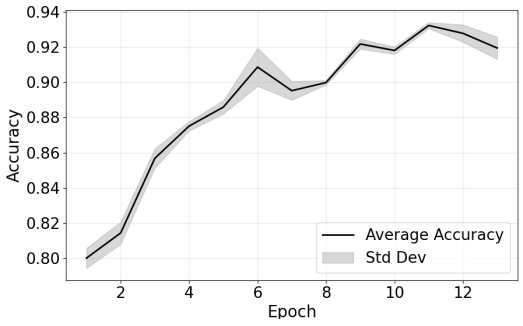


Figure 10: Average accuracy and standard deviation across three runs of the first phase of the training (illustrated in Figure 3)with noise instead of graph embeddings.

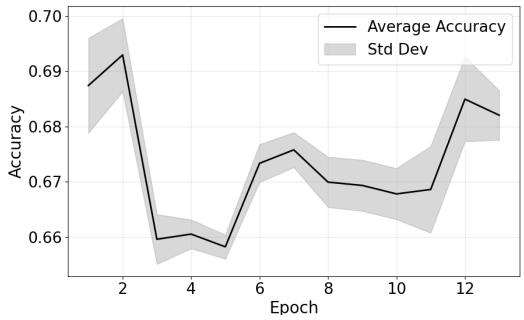


Figure 11: Average accuracy and standard deviation across three runs of the first phase of the training (illustrated in Figure 3)with noise instead of amino acid sequence embeddings.

V. Conclusion

In this study, we explored additive and multiplicative modality fusion strategies for a transformer-based algorithm aimed at predicting enzyme-substrate interactions. By extending the ProSmith architecture, we incorporated graph representations of protein structures as an additional input modality and analyzed the effects of additive and multiplicative fusion techniques, including embedding normalization and balancing weights.

Our experiments revealed that the incorporation of graph-based information does not significantly impact the performance of the model. The results suggest that the ProSmith model outperforms our implementation in random and cold protein data splits. However, we achieved improvement in the cold SMILES and cold protein & SMILES splits.

Furthermore, our model exhibited superior performance on underrepresented substrates, showcasing its potential for better generalization, particularly in datasets with limited positive samples.

A limitation of our study is the restricted number of training and testing runs. Conducting significantly more experiments and averaging the results with established confidence intervals would improve the reliability of our conclusions.

Our error analysis did not reveal any correlation between specific graph properties and higher error rates, leaving us without a straightforward explanation for the model's misclassifications. The ablation study emphasized the robustness of the multimodal transformer and its adaptability even when faced with serious input perturbations. Additionally, while the gradient boosting step contributed marginally to performance improvements, it was less impactful than anticipated.

Overall, our findings show the potential of multiplicative fusion for integrating additional modalities into a transformer-based model without retraining from scratch. Future work should focus on conducting more experimental runs to validate these results with greater confidence as well as testing higher-order equations for modeling the interactions between modalities. Additionally, efforts should go in the direction of enriching the small molecule data in the input as the algorithm struggles more with generalizing to unseen substrates as opposed to unseen proteins.

References

[1] Walid Ahmad et al. “Chemberta-2: Towards chemical foundation models”. In: *arXiv preprint arXiv:2209.01712* (2022).

[2] Maryam Bagherian et al. “Machine learning approaches and databases for prediction of drug–target interaction: a survey paper”. In: *Briefings in bioinformatics* 22.1 (2021), pp. 247–269.

[3] Helen M Berman et al. “The Protein Data Bank”. In: *Nucleic Acids Research* 28.1 (2000), pp. 235–242. DOI: 10.1093/nar/28.1.235.

[4] Oliviero Carugo and Kristina Djinović-Carugo. “Structural biology: A golden era”. In: *PLoS biology* 21.6 (2023), e3002187.

[5] Tianqi Chen and Carlos Guestrin. “Xgboost: A scalable tree boosting system”. In: *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*. 2016, pp. 785–794.

[6] Aditya Grover and Jure Leskovec. “node2vec: Scalable feature learning for networks”. In: *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*. 2016, pp. 855–864.

[7] Arian R Jamasb et al. “Graphein-a Python library for geometric deep learning and network analysis on protein structures and interaction networks”. In: *bioRxiv* (2020), pp. 2020–07.

[8] Siddhant M Jayakumar et al. “Multiplicative interactions and where to find them”. In: *International conference on learning representations*. 2020.

[9] John Jumper et al. “Highly accurate protein structure prediction with AlphaFold”. In: *nature* 596.7873 (2021), pp. 583–589.

[10] Alexander Kroll, Sahasra Ranjan, and Martin J Lercher. “Drug-target interaction prediction using a multi-modal transformer network demonstrates high generalizability to unseen proteins”. In: *bioRxiv* (2023), pp. 2023–08.

[11] Alexander Kroll et al. “A general model to predict small molecule substrates of enzymes based on machine and deep learning”. In: *Nature communications* 14.1 (2023), p. 2787.

[12] Zeming Lin et al. “Evolutionary-scale prediction of atomic-level protein structure with a language model”. In: *Science* 379.6637 (2023). Earlier versions as preprint: bioRxiv 2022.07.20.500902, pp. 1123–1130. DOI: 10.1126/science.ade2574. URL: https://www.science.org/doi/abs/10.1126/science.ade2574.

[13] Tiqing Liu et al. “BindingDB: a web-accessible database of experimentally determined protein–ligand binding affinities”. In: *Nucleic acids research* 35.suppl_1 (2007), pp. D198–D201.

[14] Scott Reed et al. “A generalist agent”. In: *arXiv preprint arXiv:2205.06175* (2022).

[15] Chameleon Team. “Chameleon: Mixed-modal early-fusion foundation models”. In: *arXiv preprint arXiv:2405.09818* (2024).

[16] Michele Vendruscolo et al. “Small-world view of the amino acids that play a key role in protein folding”. In: *Physical Review E* 65.6 (2002), p. 061910.

[17] Christopher A Voigt. “Synthetic biology 2020–2030: six commercially-available products that are changing our world”. In: *Nature Communications* 11.1 (2020), pp. 1–6.

[18] David Weininger. “SMILES, a chemical language and information system. 1. Introduction to methodology and encoding rules”. In: *Journal of Chemical Information and Computer Sciences* 28.1 (1988), pp. 31–36. DOI: 10.1021/ci00057a005.

[19] Torsten Wörtwein et al. “Beyond additive fusion: Learning non-additive multimodal interactions”. In: *Findings of the Association for Computational Linguistics: EMNLP 2022*. 2022, pp. 4681–4696.

[20] Shuke Wu et al. “Biocatalysis: enzymatic synthesis for industrial applications”. In: *Angewandte Chemie International Edition* 60.1 (2021), pp. 88–119.

[21] Yue Zhang et al. “A survey of drug-target interaction and affinity prediction methods via graph neural networks”. In: *Computers in Biology and Medicine* 163 (2023), p. 107136.

Appendix

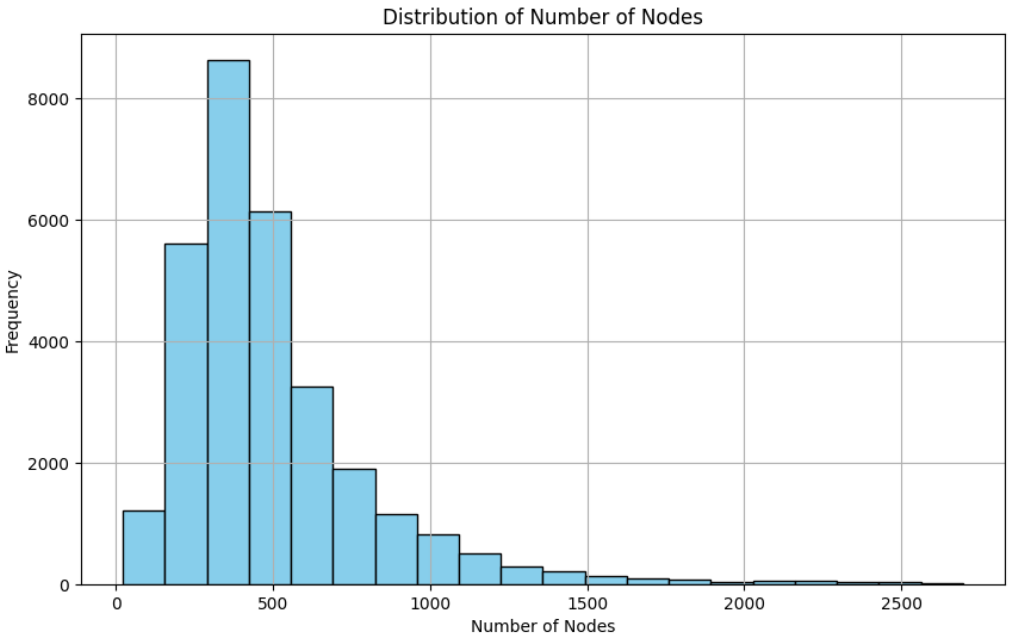


Figure A.1: Distribution of the number of nodes in the protein graphs in our dataset.

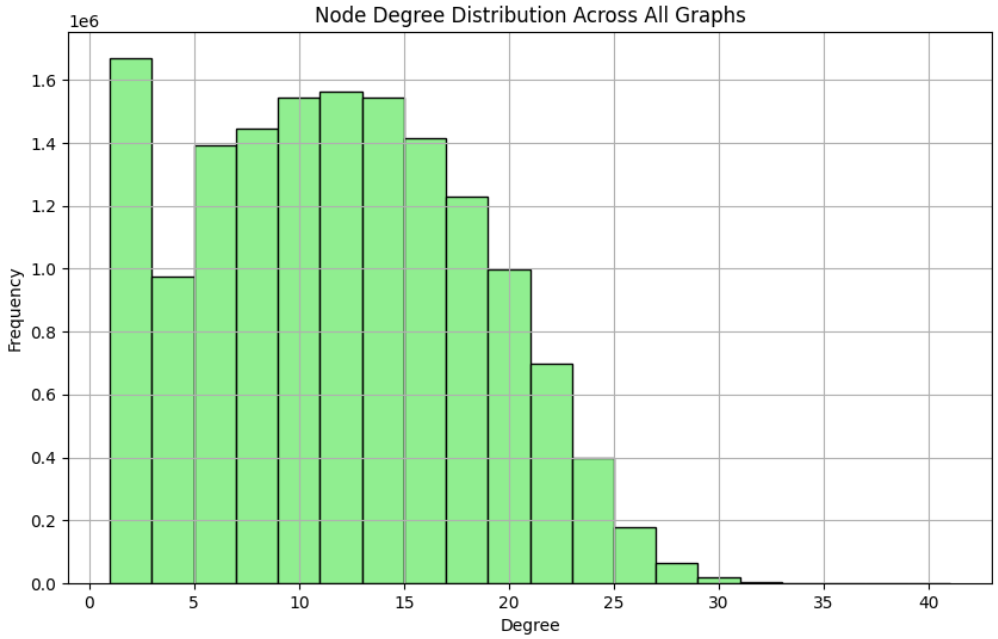


Figure A.2: Node degree distribution of the protein graphs in our dataset.

	Num Nodes	Num Edges	Density	Average Degree	Diameter
Mean	477.60	2788.73	0.03	11.60	33.55
Std	362.54	2788.73	0.02	2.89	42.78
Min	30.00	67.00	0.00	2.18	5.00
Max	2576.00	17166.00	0.15	16.22	289.00

Table A.1: Summary statistics for the graph properties of the misclassified proteins in the test set of the random split, when using the multiplicative fusion with normalized embeddings model.