Advancing Antibiotic Resistance Classification with Deep Learning Using Protein Sequence and Structure

Aymen Qabel, Sofiane Ennadir, Giannis Nikolentzos, Johannes Lutzeyer, Michail Chatzianastasis, Henrik Bostrom, Michalis Vazirgiannis

Structure-Aware Antibiotic Resistance Classification Using Graph Neural Networks

Aymen Qabel[1]*, **Sofiane Ennadir**[2], **Giannis Nikolentzos**[1], **Johannes F. Lutzeyer**[1], **Michail Chatzianastasis**[1], **Henrik Boström**[2], **Michalis Vazirgiannis**[1]

[1]DaSciM, LIX, École Polytechnique, Institut Polytechnique de Paris, France.
[2]KTH Royal Institute of Technology, Sweden.
{qabel, nikolentzos, mvazirg}@lix.polytechnique.fr
{johannes.lutzeyer, michail.chatzianastasis}@polytechnique.edu
{ennadir, bostromh}@kth.se

Abstract

Antibiotics are traditionally used to treat bacterial infections. However, bacteria can develop immunity to drugs, rendering them ineffective and thus posing a serious threat to global health. Identifying and classifying the genes responsible for this resistance is critical for the prevention, diagnosis, and treatment of infections as well as the understanding of its mechanisms. Previous methods developed for this purpose have mostly been sequence-based, relying on the comparison to existing databases or machine learning models trained on sequence features. However, genes with comparable functions may not always have similar sequences. Consequently, in this paper, we develop a deep learning model that uses the protein structure as a complement to the sequence to classify novel Antibiotic Resistant Genes (ARGs), which we expect to provide more useful information than the sequence alone. The proposed approach consists of two steps. First, we capitalize on the much-celebrated AlphaFold model to predict the 3D structure of a protein from its amino acid sequence. Then, we process the sequence using a transformer-based protein language model and apply a graph neural network to the graph extracted from the structure. We evaluate the proposed architecture on a standard benchmark dataset on which we find it to outperform state-of-the-art methods.

1 Introduction

Humans and bacteria have a long-standing symbiotic relationship which dates back to the dawn of humanity. Some bacteria are very useful for humans since they play a key role in some functions. However, there also exist bacteria which are responsible for certain human diseases such as meningitis and tuberculosis. Since 1928 when Alexander Fleming discovered penicillin, antibiotics have served as the main weapon to treat and prevent bacterial infection. Antibiotics have thus contributed to an increase in life expectancy and to the improvement of healthcare systems since surgeries and other types of operations can now be carried out safely without the fear of any bacterial infections [28, 13].

However, due to the overuse and misuse of antibiotics, bacteria become resistant and no longer respond to these drugs, a phenomenon known as antibiotic resistance. These resistant bacteria cause infections that are harder to treat resulting in higher medical costs and increased mortality. In addition to the overuse of antibiotics, the rise of antibiotic resistance is also due to the inability of the pharmaceutical industry to develop new drugs, which is due to challenging regulatory requirements

*Contact author

NeurIPS 2022 AI for Science Workshop.

among others. Antibiotic resistance is estimated to be responsible for over 33,000 deaths across Europe in 2015 [8], and if not addressed, it is expected that by 2050, around 10 million people globally could be at risk every year, with an economic cost exceeding $100 trillion [25].

Thus, it becomes clear that there is an urgent need for actions to prevent the antibiotic resistance crisis. Some examples of such actions include reducing the unnecessary use of antibiotics and using optimal treatments to cure infections. Antibiotic resistance usually occurs as the result of genetic mutations. However, bacteria can also become resistant to antibiotics by the acquisition of specific genes through horizontal gene transfer [4]. Therefore, finding the antibiotic resistant genes (ARGs) along with the antibiotic class they are resistant to is of paramount importance for the development of targeted treatment, but also for the prevention of infections.

To that end, the research community has resorted to computational methods, among others, which can compare sampled genome sequences against sequences that exist in reference databases. These methods typically employ well-established sequence alignment algorithms such as BLAST [2] and DIAMOND [7]. Most of these methods use existing microbial resistance databases along with a "best hit" approach to predict whether a sequence is indeed an ARG.

While the aforementioned approaches can accurately recognize known or highly conserved sequences of ARGs, they may fail to detect novel ARGs or those that exhibit low levels of similarity to known ARGs. This results into a significant amount of false negative samples, i.e., several ARGs are classified as not being ARGs [3, 35, 23]. Recently, some deep learning approaches have been proposed to deal with the above limitations. For example, DeepARG [3] generates features that capture the distances of a given sequence to known ARGs and passes them onto a deep learning architecture. This method was shown to improve the classification performance and to decrease the false negative rate. Hamid and Friedberg investigated the use of transfer learning in the ARG classification setting [14]. They first constructed a new dataset, called COALA (COllection of ALl Antibiotic resistance gene databases), by combining 15 different datasets, and then pre-trained an LSTM-based language model using the ULMFit (Universal Language Model Fine-tuning) method [16] on 734,848 bacterial proteins sampled from the UniRef50 database [30]. The model was subsequently trained on the newly created COALA dataset where it outperformed alignment-based methods. Another recent method, ARG-SHINE [33], integrates multiple techniques in an ensemble. More specifically, it aggregates the results of a convolutional neural network applied to the sequences, a protein functional-based logistic regression model, and a BLAST-based model. The ARG-SHINE ensemble was found to outperform the aforementioned approaches in the ARG classification task.

Despite the success of these deep learning approaches in the ARG classification task, they all operate on the genome sequence and compare patterns present in that sequence against those of a reference database. However, resistant genes can have dissimilar sequences, but might share similar structures. For instance, the fourth mobile sulfonamide resistance genes have less than 33% similarity to previously known resistance genes, while their protein structures are more than 90% similar to those of known resistance genes [26]. This is because genome evolution accumulates mutations on different bacterial genomes causing dissimilarity in the identified genes. At the same time, natural selection maintains protein structure to produce the same necessary function.

However, resolving the structure of a protein is a difficult, time- and cost-intensive task. It requires purifying the protein in the bacterial host and using specialized equipment to determine the tertiary structure through an extensive trial and error approach. This has a severe impact on the number of proteins whose structure has been resolved. More specifically, even though millions of proteins have been sequenced over the past years, only the structures of some thousands of those proteins have been resolved so far. Last year saw a breakthrough in protein structure prediction where the release of the AlphaFold II model [19] enabled the accurate prediction of proteins' 3D structures within only a few minutes.

The success of AlphaFold motivated the development of other approaches designed to predict diverse protein properties. Most of these approaches capitalize on recent advances in natural language processing and in particular on large pre-trained transformer-based language models. These models were adapted to the setting of proteins and were trained in a self-supervised manner on the millions of available protein sequences. Examples of such models include ProtTrans [9] and ProteinBERT [6]. It has been shown that these models go beyond simple pattern matching to perform higher-level reasoning and achieve state-of-the-art performance in many residue-level and protein-level tasks [6, 9, 22]. ESM2 [22] is another recent approach which can be combined with the structure module of

AlphaFold [19] and predict the 3D structure of a protein at the resolution of individual atoms with high accuracy.

While these models can develop evolutionary knowledge about proteins, they do not explicitly capture the interactions between amino acids that are in close proximity in the structure space. Therefore, introducing structural bias to these models might potentially lead to performance improvements. This is the main contribution of this paper. More specifically, we propose a general method to classify ARGs that follows the following steps. First, given a genome sequence, we use the AlphaFold II model to predict its 3D structure. Then, we use both the sequence and the structure to predict the class of the ARG. The sequence is processed by a language model, and the structure by a graph neural network (GNN) [34], a model class that has recently achieved great success in the graph representation learning community. The employed GNN is responsible for capturing the interactions between amino acids, and is integrated in the language model. The proposed method is evaluated on a standard ARG classification dataset. Our results indicate that the proposed method either outperforms or achieves performance comparable with that of state-of-the-art approaches.

The remainder of this paper is organized as follows. Section 2 introduces some preliminary concepts and gives details about AlphaFold and GNNs. Section 3 provides a detailed description of the proposed method for ARG classification. Section 4 evaluates the proposed method on the COALA dataset and finally, Section 5 concludes this work.

2 Preliminaries

In this section, we first provide details about the AlphaFold II model, define our notation and introduce message passing neural networks, the main family of GNNs.

2.1 AlphaFold II

AlphaFold II [19] has been proposed as part of the CASP14 (Critical Assessment of Protein Structure Prediction) competition. It is a deep neural network model that exploits the information in multiple sequence alignments (MSAs) of related proteins as the raw input features for end-to-end training. This model can predict the 3D atomic coordinates of folded protein structures with an accuracy mostly within the error margin of experimental structure determination methods.

The AlphaFold II model starts by employing multiple sequence alignments (MSA) with different regions weighted by importance. It then uses a module called Evoformer, to extract information about interrelationships between protein sequences and template structures. The structure module treats the protein as a residue gas moved around by the network to generate the protein's 3D structure followed by local refinement to provide the final prediction.

In our work, since experimentally defined structures for the majority of the proteins in our dataset are not available, we utilize AlphaFold II to approximate them.

2.2 Graph Neural Networks

Notation. Let $G = (V, E)$ be an undirected graph consisting of a set of nodes V and a set of edges E. We will denote the number of nodes by n and the number of edges by m. The neighborhood of a node $v \in V$ is the set of adjacent nodes of v and is denoted by $\mathcal{N}(v)$. The degree $\deg(v)$ of a node v is equal to the number of neighbors of v, i.e., $\deg(v) = |\mathcal{N}(v)|$. The adjacency matrix $\mathbf{A} \in \mathbb{R}^{n \times n}$ of a graph G is a symmetric matrix used to encode edge information in the graph, where the ij-th element is equal to the weight of the edge between the i-th and j-th node in the graph if such an edge exists, and 0 otherwise.

Message passing neural networks. GNNs are neural network architectures that operate on graph-structured data. The first GNNs were proposed several years ago [29, 27], but these models attracted significant attention very recently [12, 34]. Most GNNs belong to the family of message passing neural networks [12]. Such models employ a message passing procedure which uses both the graph structure and the nodes' features to generate new node representations. Specifically, message passing models iteratively update the representation of each node by aggregating the representations of its

neighbors and by combining them with the node's previous representation. In other words, a message passing GNN updates the nodes' feature vectors by aggregating local neighborhood information.

Formally, suppose we have a GNN model that contains T neighborhood aggregation layers. Let also $\mathbf{h}_v^{(0)}$ denote the initial feature vector of node v, i. e., the row of matrix $\mathbf{X}$ that corresponds to node v. In each iteration ($t > 0$), the hidden state $\mathbf{h}_v^{(t)}$ of node v is updated as follows:

$$\begin{aligned} \mathbf{a}_v^{(t)} &= \text{AGGREGATE}^{(t)}\Big(\big\{\mathbf{h}_u^{(t-1)} \colon u \in \mathcal{N}(v)\big\}\Big), \\ \mathbf{h}_v^{(t)} &= \text{COMBINE}^{(t)}\Big(\mathbf{h}_v^{(t-1)}, \mathbf{a}_v^{(t)}\Big), \end{aligned} \tag{1}$$

where AGGREGATE is a permutation invariant function that maps the feature vectors of the neighbors of a node v to an aggregated vector. This aggregated vector is passed along with the previous representation of v, i. e., $\mathbf{h}_v^{(t-1)}$, to the COMBINE function which combines the two vectors to produce the new representation of v. After T iterations of neighborhood aggregation, to produce a graph-level representation, GNNs apply a permutation invariant readout function (e. g., the sum operator or mean operator) to the feature vectors of all nodes of the graph as follows:

$$\mathbf{h}_G = \text{READOUT}\Big(\big\{\mathbf{h}_v^{(T)} \colon v \in V\big\}\Big). \tag{2}$$

3 Methodology

In this section, we present the proposed approach for the ARG classification problem. More specifically, the proposed method consists of three main steps: (1) AlphaFold II is first employed to predict the protein structure of the input amino acid sequences; (2) the emerging protein structures are then mapped to graph structures; and (3) the generated graphs are fed into a message passing GNN which in turn is integrated into a large-scale pre-trained protein language model. An overview of the proposed method is illustrated in Figure 1. We next give more details about each one of the three steps of the proposed method.

3.1 From Sequences to Structures

We start by predicting the 3D structure of the different proteins in our dataset. Due to the size of the dataset and limited resources, we made use of ColabFold [24], a software that replaces the MSA generation part of AlphaFold II with an alternative one that is faster and thus reduces the prediction time, which can take many minutes for one protein.

AlphaFold II enhances the predicted protein structure by recycling (by default) three times, meaning that the prediction is fed multiple times through the model. AlphaFold II also computes five models through multiple recycles. Given that this can be time-consuming, we limited our predictions to the be the first with an average per-residue confidence metric (pLDDT) higher than 70%. AlphaFold II produces this pLDDT score to measure its confidence in the predicted structure, where higher scores indicate better predictions. In [10] the authors mention that predictions above the threshold of 70% pLDDT can be expected to be modelled well.

3.2 From Structures to Graphs

The main advantage of our proposed method compared to previously proposed approaches is that the predictive model, besides sequence, also takes structure into account. Thus, both sequence and structure are leveraged to determine the resistance class. Following previous studies [5, 21], in this work, we represent the 3D structure as a graph. More specifically, for each protein sequence, AlphaFold II outputs a PDB file which stores the 3D coordinates of the atoms in the different amino acids. Then, we build a graph $G = (V, E)$ as follows: Each node $v \in V$ represents an amino acid that exists in the protein, and is associated with a feature vector $\mathbf{x}_v = \mathbf{h}_v^{(0)}$. Each edge $e \in E$ connects two amino acids to each other. In our setting, there are two types of edges: (1) peptide bonds which connect residues that are adjacent in the primary sequence; and (2) "proximity" edges that connect

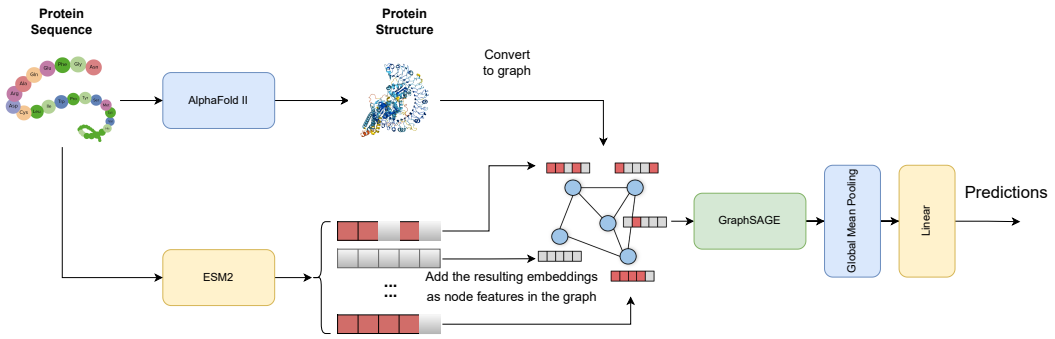

Figure 1: High-level overview of the proposed framework for classifying the different ARGs. The framework consists of two main components : (1) The AlphaFold model predicting the 3D structure of an input protein; (2) A classifier, consisting of a GraphSAGE model applied to the graph constructed from the structure and with node features from a pretrained protein language model (ESM2) applied to the sequence.

(non-bonded) residues which are spatially close to each other. For the second type of edges, we only consider pairs of residues whose distance is less than a pre-defined threshold ϵ usually chosen between 6.5 Å and 9.5 Å in the relevant literature [17]. In this work, we used a threshold of 8 Å given that it yields the best validation accuracy results. Since edges capture distance relationships between nodes, we choose to build a weighted graph. We use the following weighting scheme for the edges:

$$\mathbf{A}_{ij} = \begin{cases} 1 - \frac{d(\mathbf{x}_i, \mathbf{x}_j)}{\epsilon+1} & \text{if } d(\mathbf{x}_i, \mathbf{x}_j) < \epsilon, \\ 0 & \text{otherwise,} \end{cases} \tag{3}$$

where $d(\mathbf{x}_i, \mathbf{x}_j)$ is the Euclidean distance between the coordinates of the corresponding C_α atoms of the i-th and j-th amino acid, and ϵ equals the pre-defined threshold of 8 Å. Note that edge weights typically represent the similarity between the edge's endpoints, and therefore we transform distances into similarities. We should also mention that not all GNNs take edge weights as input, and for those GNNs we can set the weight of all edges equal to one. Several different approaches were employed to generate the node feature vectors which will be discussed in the forthcoming Section 3.3.

3.3 Classification Model

We propose to use a message passing GNN to capture the structural information of proteins. The GNN is applied to the graphs that emerge from 3D structures as described in Section 3.2. To allow the model to learn more complex patterns, we annotate the nodes of the graph with embeddings from a pre-trained language model (ESM2 in our case). Thus, each node $v \in V$ is assigned an initial feature vector $\mathbf{h}_v^{(0)}$ that contains this embedding. While pre-trained language models have shown a high learning ability of the internal representations of proteins and are capable of capturing some functional and structural information, they can still benefit from gaining explicit access to the structure since it can more accurately capture the interactions between amino acids that are in close proximity.

To generate the amino acid embeddings, we employ the Evolutionary Scale Modeling (ESM2) model [22]. ESM2 is the largest pre-trained protein language model to date. It mainly consists of transformer layers and was trained on over 60 million protein sequences. It has shown very promising results in predicting the 3D structure of proteins when followed by the structure module of AlphaFold [19], without using multiple sequence alignments. This indicates that the amino acid embeddings it produces are semantically rich and implicitly contain evolutionary information. Note that there exist several ESM2 variants available which usually differ in the number of parameters, ranging from 8 million parameters to 15 billion parameters. Due to limited resources, we only use the model that consists of 8 million parameters.

As already mentioned, the ESM2 amino acid embeddings correspond to the initial features of the nodes and are then transformed using the GraphSAGE model [15]. The AGGREGATE function of

GraphSAGE is defined as follows:

$$\mathbf{h}_v^{(t+1)} = \sigma\left(\mathbf{W}^{(t)} \frac{\mathbf{h}_v^{(t)} + \sum_{u \in \mathcal{N}(v)} \mathbf{h}_u^{(t)}}{\deg(v) + 1}\right),$$

$$\mathbf{h}_v^{(t+1)} = \frac{\mathbf{h}_v^{(t+1)}}{\left\|\mathbf{h}_v^{(t+1)}\right\|_2},$$

where $\mathbf{W}^{(t)}$ is a matrix of trainable parameters and $\sigma(\cdot)$ a non-linear activation function. Finally, to produce a representation for an entire graph, we apply the mean operator to the feature vectors of all nodes. Thus, our READOUT function is defined as:

$$\mathbf{h}_G = \frac{1}{n} \sum_{v \in V} \mathbf{h}_v^{(T)}, \tag{4}$$

where T is the number of message passing iterations. The graph representation $\mathbf{h}_G$ is further fed into a linear layer to produce the class probabilities.

4 Experimental Evaluation

We experimentally evaluate the proposed method in the ARG classification task. We begin by describing the benchmark dataset, the baseline methods and the experimental setup. Finally, we report the empirical results and discuss the model's performance and limitations.

4.1 Experimental Details

Dataset. We evaluate the proposed method on the publicly available benchmark dataset "COllection of ALl Antibiotic resistance gene databases", commonly referred to as COALA [14]. The dataset agglomerates a number of available databases and provides the protein sequences and their corresponding resistance classes. We perform CD-HIT [11] clustering of sequences with a 70% threshold to have a dataset with maximum 70% identity between sequences. The identity score between sequences refers to the ratio of identical amino acids in the aligned sequences. We ended up with $11{,}090$ proteins, each belonging to one of 16 different classes of antibiotic resistance (see Appendix B Table 3 for more details). The number of nodes and edges of the generated graphs depend on the considered sequence and threshold ϵ. On average, the graphs consist of 281.05 nodes and 1624.94 edges and the average degree of nodes corresponds to 5.71.

Experimental setup. The different samples were split into training and test sets with an $80:20$ split ratio. The test set was created in such a way that the test sequences have at most 40% identity with the train sequences. To that end, we once again used CD-HIT [11] to define the various clusters. The minimum "in-identity" between sequence pairs within the same cluster is 40%, whereas the maximum "out-identity" between sequence pairs within different clusters is 40%. We iteratively add clusters to the training and test sets, ensuring a stratified split. The whole process was repeated five times giving rise to five splits. Those splits are different from each other, something that is not trivial to achieve since there are some large clusters of sequences in the dataset which need to be included in the training set. As a result, we have overlapping (but not identical) test sets across the different splits as seen in Appendix A Figure 2.

Training and hyperparameters. We train the full model, consisting of the ESM2 model followed by the GNN model on the COALA dataset, using the Adam optimizer with a learning rate equal to $3 \cdot 10^{-4}$ trying to minimize the cross-entropy loss. We experimented with different message passing GNN architectures (e.g., Graph Convolutional Network [20], Graph Attention Network [31] and GraphSAGE [15]) and we report the results of the best-performing model (GraphSAGE in our setting). We experimented with values of threshold ϵ from 5 Å to 15 Å and report the best results (8 Å). We used a dropout rate of 0.2 and a batch size of 32. Our graph neural network has 4 hidden layers with a hidden dimension of 64. We used ReLU activations and batch normalization [18] between each layer of the GNN.

Table 1: Classification results (± standard deviation) of the different approaches on the COALA dataset. The best performance in the different groups is typeset in **bold**. Note that N/A refers to "not applicable".

METHOD	ACCURACY	MACRO F1-SCORE	ACCURACY (<50% ID)	ACCURACY (>50% ID)	ACCURACY (HOMOLOG NOT FOUND)
BLAST	65.42% (± 0.57%)	59.29% (± 1.15%)	75.11% (± 1.78%)	**95.29% (± 1.18%)**	N/A
DIAMOND	58.86% (± 0.62%)	56.46% (± 2.56%)	64.69% (± 1.72%)	95.11% (± 1.53%)	N/A
TF-IDF	54.19% (± 1.62%)	42.18% (± 2.40%)	55.18% (± 1.72%)	84.05% (± 2.07%)	35.43% (± 1.25%)
TRAC	69.80% (± 0.66%)	59.70% (± 1.73%)	72.78% (± 1.77%)	90.92% (± 2.16%)	36.83% (± 4.36%)
ARG-SHINE	68.34% (± 1.27%)	58.21% (± 2.84%)	69.60% (± 1.78%)	92.88% (± 0.70%)	38.00% (± 3.88%)
ARGGNN	**72.90% (± 0.65%)**	**63.78% (± 1.15%)**	**76.49% (± 1.30%)**	93.00% (± 1.10%)	**38.90% (± 3.98%)**

Evaluation metrics. Our main evaluation metrics are accuracy and macro f1-score. Besides the overall accuracy, we also report the accuracy with respect to the identity with the best hit in the training set. Specifically, during the evaluation, we split the test set into three different groups: one that represents the elements that have a best hit homolog in the training set with more than 50% identity using BLAST [2], another set for those with less than 50% identity and the last set with proteins with no homolog found by BLAST in the training set. This will allow us to test our model on ARGs that are similar to the training sequences, as well as to examine the generalization ability of our model, by testing it on novel sequences that are very different from the training samples.

Baselines. Our list of baselines includes mainly homology and sequence-based approaches. Specifically, we compare the proposed method against the following 5 baseline approaches:
(1) **BLAST** [2] is one of the most powerful tools for sequence alignment, which is used by most alignment-based methods. We ran it with max_target_seqs = 1 and different e-value cut-offs (10^{-3}, 10^{-7}, 10^{-10}) as the representatives of the best hit approach.
(2) **DIAMOND** [7] is another tool for sequence alignment. It is faster than BLAST, but less accurate.
(3) **TF-IDF** [1] is a method originally applied in the field of natural language processing to map textual documents to vectors. We generate n-gram features for different values of n and feed these features to a random forest classification model to make predictions.
(4) **TRAC** [14] employs a transfer learning technique and follows the ULMFit training strategy [16]. The pre-trained TRAC model (LSTM-based language model) was fine-tuned on the COALA dataset with a language model objective, and then fine-tuned for a second time on the classification task.
(5) **ARG-SHINE** [32] is an ensemble method that combines the output of BLAST along with a convolutional neural network applied to sequences, and information about the family, domain and motif of proteins. We compare only against their convolutional neural network component since it is the model's dominant component, while the ensemble is based mainly on BLAST results, which can be used with all other models to improve their performance.

4.2 Results and Discussion

In Table 1 we report the resulting average prediction accuracy and the corresponding standard deviation of each experiment. Our model (ARGGNN) yields an absolute improvement of 3.1% in macro accuracy over TRAC, the best competitor from the baselines. Moreover, our proposed framework performs better than the other benchmark methods in the first setting ($< 50\%$ id) and the third setting, where no homolog is found. Also, BLAST and DIAMOND can not make predictions if there is no homolog found in the training set. Finally, ARGGNN achieves better results than the DL-based methods in the second setting ($> 50\%$ id) while being less performant than both BLAST and DIAMOND. To improve the results even further, we can use the ensembling method of [32] and incorporate BLAST results when the identity is very high. Table 1 illustrates the superiority of the ESM + GraphSAGE model over the previous ones once more, with the Macro F1 score improving from 59.7% to 63.78%.

In summary, the results in Table 1 show the proposed method's ability to detect ARGs, even those with low similarity to the training set. Many novel ARGs contain protein sequences that differ from existing ones, and standard homology-based methods fail to detect them, resulting in a high false negative rate. In contrast, our model can effectively identify ARGs which are out of the distribution of the training set.

Table 2: Additional analysis of the utility of combing GraphSAGE and the ESM model. Classification accuracy (± standard deviation) of the different approaches on the COALA dataset. The best performance in the different groups is typeset in **bold**.

METHOD	ACCURACY	ACCURACY (<50% ID)	ACCURACY (>50% ID)	ACCURACY (HOMOLOG NOT FOUND)
ESM	70.86% (± 1.33%)	74.37% (± 1.70%)	92.60% (± 0.50%)	35.45% (± 4.93%)
GRAPHSAGE	66.67% (± 0.98%)	69.38% (± 1.68%)	89.72% (± 1.48%)	32.57% (± 3.18%)
ARGGNN	**72.90% (± 0.65%)**	**76.49% (± 1.30%)**	**93.00% (± 1.10%)**	**38.90% (± 3.98%)**

We now further investigate the effect of combining sequence and structural information. To do so, we first train a GraphSAGE model on the resistance prediction task, using as feature vectors one-hot-encoding vectors representing the amino acid type as well as the phi and psi angles (corresponding to the peptide torsion angles) and the RSA (corresponding to the relative solvent accessibility, which measures the extent of burial or exposure of that amino acid in the 3D structure). This model has only access to the structural information. We further train a model which has only access to the sequence information. Specifically, we fine-tuned the ESM Language model on the classification task, using an MLP (Multi Layer Perceptron) following the BOS (Beginning of Sequence) token embedding. We report the results in Table 2. We observe that the combined approach ESM + GraphSAGE outperforms the individual GraphSAGE and ESM2 models. This confirms our hypothesis that injecting the structural information into sequence-based methods facilitates the learning process and enhances the predictive power of such models.

It should be noted that these findings are based on the ESM2 variant with 8 million parameters, which is the smallest of all available model variants. It was created as part of research to investigate the influence of scaling language models on their structure prediction performance, and the largest models performed significantly better. We hypothesise that a larger language model, for example with 15 billion parameters, may lead to further improvements on this benchmark and defer such investigations to future work.

5 Conclusion

This work explores a new perspective to detect antibiotic resistance using both the protein structure and sequence information. We have found that injecting the structural bias into pre-trained language models on protein sequences, using a graph neural network, leads to performance improvements over the state-of-the-art approaches. In future research, we plan to experiment with larger language models, as well as to employ self-supervised strategies for pretraining the graph neural network model.

References

[1] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[2] Stephen F Altschul, Warren Gish, Webb Miller, Eugene W Myers, and David J Lipman. Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410, 1990.

[3] Gustavo Arango-Argoty, Emily Garner, Amy Pruden, Lenwood S Heath, Peter Vikesland, and Liqing Zhang. Deeparg: a deep learning approach for predicting antibiotic resistance genes from metagenomic data. *Microbiome*, 6(1):1–15, 2018.

[4] Jessica Blair, Mark A Webber, Alison J Baylay, David O Ogbolu, and Laura JV Piddock. Molecular mechanisms of antibiotic resistance. *Nature reviews microbiology*, 13(1):42–51, 2015.

[5] Karsten M Borgwardt, Cheng Soon Ong, Stefan Schönauer, SVN Vishwanathan, Alex J Smola, and Hans-Peter Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56, 2005.

[6] Nadav Brandes, Dan Ofer, Yam Peleg, Nadav Rappoport, and Michal Linial. Proteinbert: A universal deep-learning model of protein sequence and function. *Bioinformatics*, 38(8):2102–2110, 2022.

[7] Benjamin Buchfink, Chao Xie, and Daniel H Huson. Fast and sensitive protein alignment using diamond. *Nature methods*, 12(1):59–60, 2015.

[8] Alessandro Cassini, Liselotte Diaz Högberg, Diamantis Plachouras, Annalisa Quattrocchi, Ana Hoxha, Gunnar Skov Simonsen, Mélanie Colomb-Cotinat, Mirjam E Kretzschmar, Brecht Devleesschauwer, Michele Cecchini, et al. Attributable deaths and disability-adjusted life-years caused by infections with antibiotic-resistant bacteria in the eu and the european economic area in 2015: a population-level modelling analysis. *The Lancet infectious diseases*, 19(1):56–66, 2019.

[9] Ahmed Elnaggar, Michael Heinzinger, Christian Dallago, Ghalia Rehawi, Yu Wang, Llion Jones, Tom Gibbs, Tamas Feher, Christoph Angerer, Martin Steinegger, et al. Prottrans: towards cracking the language of lifes code through self-supervised deep learning and high performance computing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021.

[10] Deepmind & EMBL-EBI. Alphafold protein structure database: Frequently asked questions. `https://alphafold.ebi.ac.uk/faq`, 2022. accessed October 2022.

[11] Limin Fu, Beifang Niu, Zhengwei Zhu, Sitao Wu, and Weizhong Li. Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152, 2012.

[12] Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1263–1272, 2017.

[13] Ian M Gould and Abhijit M Bal. New antibiotic agents in the pipeline and how they can help overcome microbial resistance. *Virulence*, 4(2):185–191, 2013.

[14] Md-Nafiz Hamid and Iddo Friedberg. Transfer learning improves antibiotic resistance class prediction. *bioRxiv*, 2020.

[15] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, 2017.

[16] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 328–339, 2018.

[17] Jun Huan, Deepak Bandyopadhyay, Wei Wang, Jack Snoeyink, Jan Prins, and Alexander Tropsha. Comparing graph representations of protein structure for mining family-specific residue-based packing motifs. *Journal of Computational Biology*, 12(6):657–671, 2005.

[18] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[19] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[20] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations*, 2017.

[21] Jaechang Lim, Seongok Ryu, Kyubyong Park, Yo Joong Choe, Jiyeon Ham, and Woo Youn Kim. Predicting drug–target interaction using a novel graph neural network with 3d structure-embedded graph representation. *Journal of chemical information and modeling*, 59(9):3981–3988, 2019.

[22] Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Sal Candido, et al. Language models of protein sequences at the scale of evolution enable accurate structure prediction. *bioRxiv*, 2022.

[23] Andrew G McArthur and Kara K Tsang. Antimicrobial resistance surveillance in the genomic age. *Annals of the New York Academy of Sciences*, 1388(1):78–91, 2017.

[24] Milot Mirdita, Konstantin Schütze, Yoshitaka Moriwaki, Lim Heo, Sergey Ovchinnikov, and Martin Steinegger. Colabfold: making protein folding accessible to all. *Nature Methods*, pages 1–4, 2022.

[25] Jim O'Neill. Tackling drug-resistant infections globally: final report and recommendations. 2016.

[26] Mohammad Razavi, Nachiket P Marathe, Michael R Gillings, Carl-Fredrik Flach, Erik Kristiansson, and DG Joakim Larsson. Discovery of the fourth mobile sulfonamide resistance gene. *Microbiome*, 5(1):1–12, 2017.

[27] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

[28] Saswati Sengupta, Madhab K Chattopadhyay, and Hans-Peter Grossart. The multifaceted roles of antibiotics and antibiotic resistance in nature. *Frontiers in microbiology*, 4:47, 2013.

[29] Alessandro Sperduti and Antonina Starita. Supervised neural networks for the classification of structures. *IEEE Transactions on Neural Networks*, 8(3):714–735, 1997.

[30] Baris E Suzek, Yuqi Wang, Hongzhan Huang, Peter B McGarvey, Cathy H Wu, and UniProt Consortium. Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932, 2015.

[31] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations*, 2018.

[32] Dong Wang, Jie Li, Yadong Wang, and Edwin Wang. A comparison on predicting functional impact of genomic variants. *NAR Genomics and Bioinformatics*, 4(1), 01 2022. lqab122.

[33] Dong Wang, Jie Li, Yadong Wang, and Edwin Wang. A comparison on predicting functional impact of genomic variants. *NAR genomics and bioinformatics*, 4(1), 2022.

[34] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 32(1):4–24, 2020.

[35] Ying Yang, Xiaotao Jiang, Benli Chai, Liping Ma, Bing Li, Anni Zhang, James R Cole, James M Tiedje, and Tong Zhang. Args-oap: online analysis pipeline for antibiotic resistance genes detection from metagenomic data using an integrated structured arg-database. *Bioinformatics*, 32(15):2346–2351, 2016.

A Visualization of the Different Test Sets in a Venn Diagram.

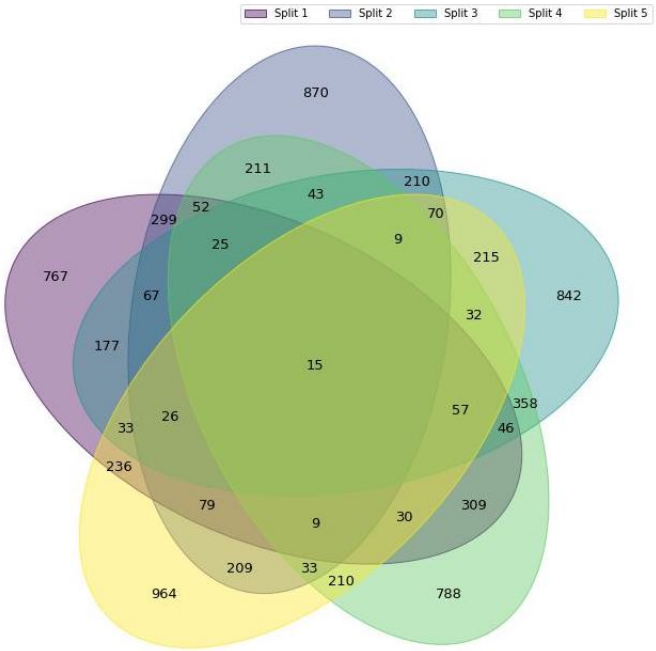

Figure 2: Venn Diagram showing the number of common elements between the five different test sets, i.e., of the different data splits.

As described in the Section 4.1, our dataset is divided into five different train-test splits, without having any overlap between the training and the test sets.

Between the test sets, there may be common sequences and this is due to the constraint of having at most 40% identity between the training and test sets. Some clusters given by CD-HIT [11] are larger than the test size and thus can only be in the training set, otherwise the constraint is not respected.

For example, each test set has an average of 846 proteins that are totally different of all the other test sets, which represents around 8% of the whole dataset and only 15 proteins are present in all the test sets as indicated by the intersection of the five large ellipses in Figure 2.

B Description of the Dataset, Along with the Splitting

Table 3: Description of the COALA Dataset, along with the splitting.

	Splits 1-5	
Antibiotic Resistance	Train	Test
BETA-LACTAM	3182	796
FOLATE-SYNTHESIS-INHABITOR	1339	335
GLYCOPEPTIDE	1268	318
TETRACYCLINE	813	204
AMINOGLYCOSIDE	771	193
TRIMETHOPRIM	342	86
MACROLIDE	323	81
PHENICOL	264	67
QUINOLONE	157	40
SULFONAMIDE	146	37
MULTIDRUG	118	31
FOSFOMYCIN	53	14
BACITRACIN	45	12
MACROLIDE/LINCOSAMIDE/STREPTOGRAMIN	16	5
STREPTOGRAMIN	14	4
RIFAMYCIN	12	4

The train-test split of the dataset is done in a stratified way, conserving the distribution of the classes in the dataset. Some classes like Rifamycin, streptogramin and MLS (macrolide/lincosamide/streptogramin) only contain a small number of training examples, which is a limitation for deep learning models. There are some methods to deal with these kind of problems. We explored using a weighted cross entropy loss, by injecting the corresponding weights for each class into the loss, but found that it didn't have a significant effect. We believe that a creation of a good quality dataset is of paramount importance to make significant progress in this field.