Transfer learning towards combating antibiotic resistance

by

Md Nafiz Hamid

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Bioinformatics and Computational Biology

Program of Study Committee:
Iddo Friedberg, Co-major Professor
Drena Dobbs, Co-major Professor
Adina Howe
Justin Walley
Gregory Phillips

The student author, whose presentation of the scholarship herein was approved by the program of study committee, is solely responsible for the content of this dissertation. The Graduate College will ensure this dissertation is globally accessible and will not permit alterations after a degree is conferred.

Iowa State University

Ames, Iowa

2019

DEDICATION

To my Parents who have sacrificed so much for me.

TABLE OF CONTENTS

Page

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGMENTS

All praises belong to God, and Peace and Blessings be upon His Messenger and final Prophet Muhammad. I would like to thank my advisor Dr. Iddo Friedberg for his unconditional support over the years, and for giving me full independence over the projects I wanted to pursue. I am immensely grateful to you, Iddo. I would also like to thank my co-advisor Dr. Drena Dobbs who guided me, and always cheered me on in this long journey. Many thanks to my committee members, Dr. Adina Howe, Dr. Justin Walley, and Dr. Gregory Phillips for keeping me on my toes in terms of my scientific goals. I will always be grateful to former co-ordinator of the Bioinformatics Graduate program, Trish Stauble. Trish, I would not have reached the end of my PhD without your unconditional love and support. I would like to thank my sister, Nazifa for being there with me, and cheering me up. Shout out to my baby boy, Musab who came into the world during my PhD, and gave me a bigger appreciation of life and its meaning. Last but not the least, I am grateful to my wife, Tamanna for being patiently by my side over these years during which I spent many days and nights occupied with work.

ABSTRACT

Transfer learning with deep neural networks has revolutionized the fields of computer vision and natural language processing in the last decade. This is especially significant for fields such as biology where we usually have small labeled data but an abundance of unlabeled data. Using abundant unlabeled data to enhance performance on a small labeled dataset is the hallmark of transfer learning. In this dissertation, I tap into the potential of transfer learning to solve critical problems in the antibiotic resistance domain. Antibiotic resistance occurs when bacteria gain functionality to thwart mechanisms through which antibiotics work to kill or inhibit bacteria. This resistance is leading to alarming rates of mortality and morbidity among the world population. Two critical aspects in combating antibiotic resistance is searching for novel sources of antibiotics, and identifying genes that confer antibiotic resistance ability to a bacteria. As I show, in both of these cases, we have small labeled datasets but large unlabeled data at our disposal. I have incorporated transfer learning techniques in both cases, significantly improving on current state-of-the-art performance typically achieved by alignment based approaches such as BLAST or HMMER. I also introduce a novel optimization method to train neural networks that offer reliable uncertainty estimates when the model is tested on Out-of-distribution (OoD) data. Finally, I offer future directions on how transfer learning can be further utilized to solve these critical problems.

CHAPTER 1. INTRODUCTION

Antibiotics are drugs used to treat bacterial infections. Since their introduction during the 1940s, they have saved millions of human lives (Sengupta et al., 2013). Antibiotics have played a critical role in increasing life expectancy which has increased from 56.4 years to almost 80 years (Shrestha et al., 2005) in the United States. These drugs also help people with chronic diseases such as diabetes, renal diseases as well as people undergoing major surgery survive life threatening infections (Ventola, 2015). Yet antibiotic resistance – the phenomenon when previously treatable microbes gain the ability to resist antibiotics and survive – is threatening to ruin all medical progress in today's world. Overuse of antibiotics, over-zealous prescription of antibiotics, extensive use in agriculture, release of antibiotics into environment without proper wastewater treatment during production, patients not finishing their course etc are making the situation worse (Read and Woods, 2014; for Disease Control and Prevention, 2018). In this chapter, I provide a brief history of antibiotics and antibiotic resistance, an outline of problems I try to solve using machine learning, and an introduction to such machine learning methods, and how this dissertation is structured.

1.1 Antibiotics and Antibiotic resistance

The term antibiotics, in its strict usage, refers to agents produced by microorganisms that kill or inhibit other microorganisms. The term antimicrobials refers to a broader set of agents that can include antibiotics as well as synthetic, semi-synthetic or natural agents originating from plants and animals. Unlike antibiotics, antimicrobials may also inhibit or kill viruses, fungi or protozoa along with bacteria. The term antibacterials is also often used interchangeably with the term antimicrobials.

Antibiotics kill or inhibit growth of microbes by different mechanisms of targeting bacterial physiology and biochemistry (Sengupta et al., 2013), and can be classified by their target site.

For example, β-lactams, Glycopeptides, Fosfomycins attack cell wall formulation. Sulfonamides and Trimethoprim disrupt nucleic acid synthesis. Macrolides, Streptpgramins, Amphenicol, Lincosamide etc attack protein synthetic machinery by targeting ribosomal subunits (Boolchandani et al., 2019). A few antibiotics like Lipopeptide and Polymyxin target cell membranes. Antibiotic resistance happens when bacteria circumvent the mechanisms described above. More specifically, resistance instantiate through reduced permeability, antibiotic efflux, antibiotic target site protection, and many other ways (Boolchandani et al., 2019).

Antibiotics and antibiotic resistance have developed mutually because of the production of antibiotics in the biosphere by microorganisms, albeit the concentration is much lower than what is used for therapeutic purposes (Sengupta et al., 2013). Use of these secondary metabolites as antibiotics by bacteria has naturally made some of them inherently resistant towards certain antibiotics (Nicolaou and Rigol, 2018). Hence, there are resistance genes in these bacteria that confer the functionality of antibiotic resistance upon them. However when discussing the problem of antibiotic resistance that is currently plaguing the world, we largely refer to the ability of a bacteria to gain resistance against an antibiotic when it was previously susceptible to it. This can happen by gaining resistance genes from another closely related strain or another species of bacteria. Figure 1.1 shows a brief history of antibiotics introduction for therapeutic use and subsequent development of antibiotic resistance.

Figure 1.1 shows that whenever an antibiotic has been introduced, eventually resistance to it has resulted in some bacteria. At the same time, after the 19880s, the antibiotic pipeline has largely dried up. Hence, while development of resistance to antibiotics in some bacteria is natural, overuse in human prescription as well as in agricultural use have led to many resistant bacteria that are not susceptible to any of the antibiotics currently available.

Figure 1.2 shows a chart of number of antibiotics approved over the years. The number has gradually decreased over the last three decades. There are three main obstacles towards developing new antibiotics — regulatory barriers, fewer economic incentives, and scientific obstacles (Wright, 2014). Compared with other drug trials, antibiotics drug trials are significantly harder and expen-

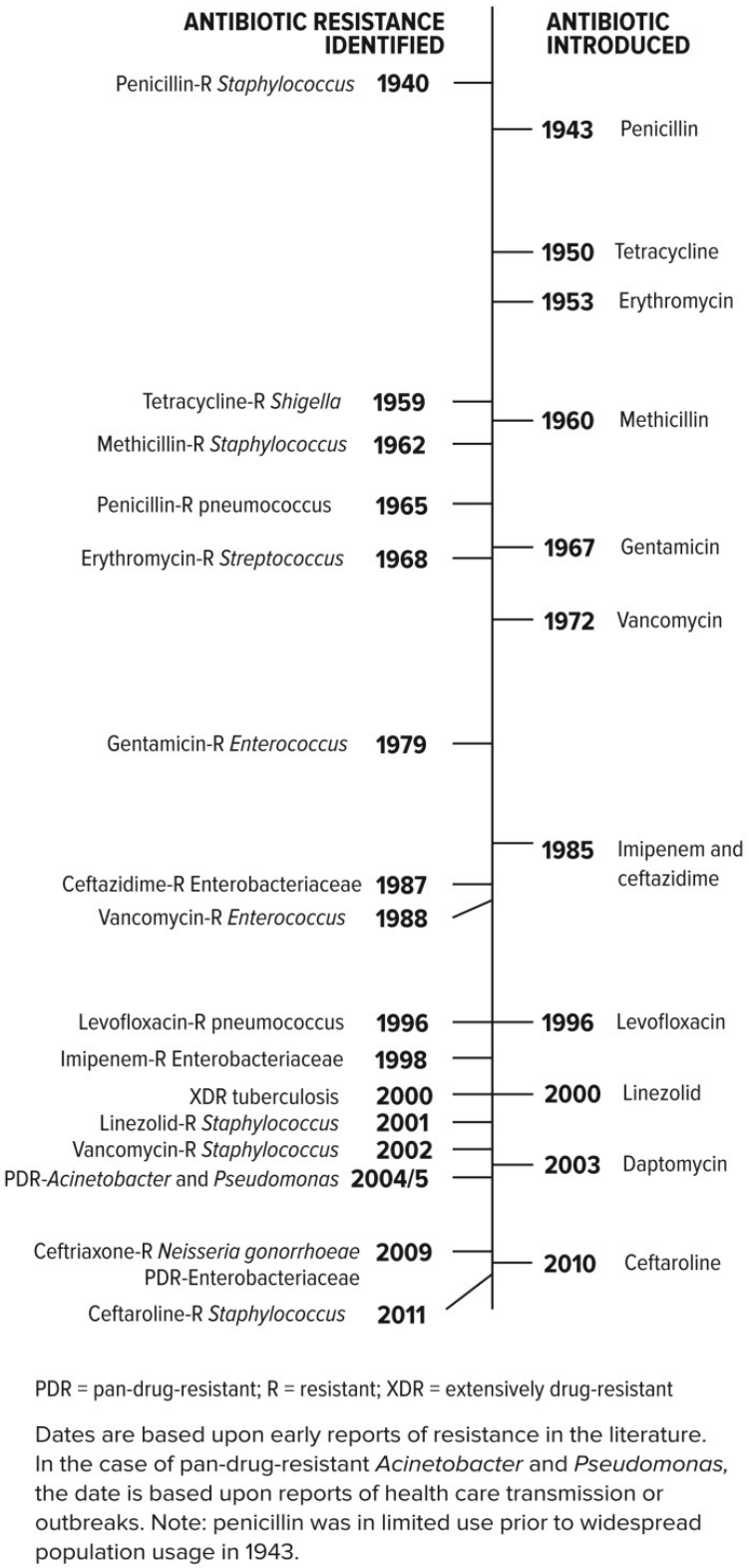


Figure 1.1: Brief history of antibiotic introduction and subsequent development of antibiotic resistance. Image from (Ventola, 2015).

sive as they require a large sample population (Piddock, 2012). Add to that complex bureaucracy, lack of clarity, and communication in pursuing regulatory approval, and many pharmaceutical companies have totally abandoned antibiotics development. Another major factor is the short term economic profit from a new antibiotic. Compared to drugs for chronic diseases which are taken for years, antibiotics cure diseases after a relatively short course. The low return on investment means pharmaceutical industry is less inclined to develop novel antibiotics. Above all, the scientific challenges in novel antibiotic discovery are not insignificant. Even though clinically used antibiotics have been traditionally from natural products which are genetically encoded small molecules, synthetic chemical scaffolds constitute the majority of prescribed drugs in the twenty-first century (Wright, 2014). In response, bacteria cells have evolved to grow resistance to these antibiotics through various mechanisms such as resistance to penetration. Natural products, in contrast, are also produced through selection that chooses for the ability to penetrate bacteria cells. With the latest whole genome sequencing techniques, and latest bioinformatics tools, there have never been a better time to look for natural products that might provide novel scaffolds for antibiotic production. In this dissertation, I have developed computational tools for two specific problems - 1) search for natural products as sources of antibiotics, and 2) predict functional antibiotic resistance genes.

1.2 Search for natural products

As mentioned, while antibiotic resistance increases, we have seen little introduction of novel antibiotics (Ventola, 2015). Therefore, we must look for novel sources of antibiotics. One potential source can be natural products genetically encoded by bacteria (Wright, 2014). These natural products have the potential to become resistant to bacteria at a much slower rate. As a result, they may lead to novel antibiotics which will function longer against their targeted bacteria. One such category of natural products is bacteriocins. Bacteriocins are ribosomally synthesized peptides produced by bacteria to kill bacteria from another strain or closely related species. Bacteriocins have the potential to provide chemical scaffolds that can be the base for producing novel antibacterials, and because of their narrow spectrum, they have the potential to have a much longer pharmaceutical

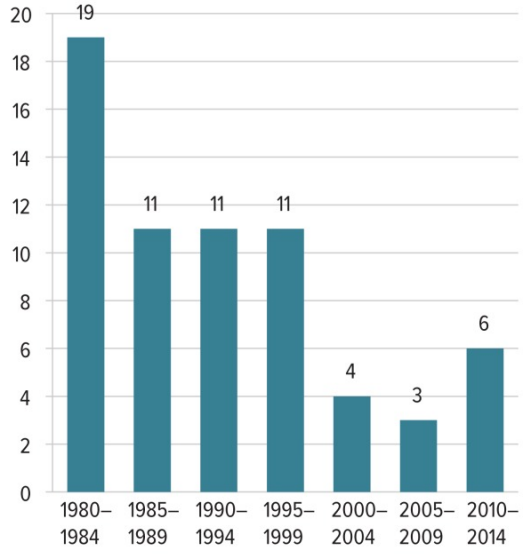


Figure 1.2: Number of antibiotics approved over the years. There has been a steady decline. Image from (Ventola, 2015).

shelf life. With the large amount of genomic data available today, the only limit in terms of finding novel bacteriocins is our ability to comb through these data. In Chapter 2, I talk about the current state-of-the-art computational tools for finding novel bacteriocins, and propose a novel machine learning method that is more efficient than these tools.

1.3 Prediction of antibiotic resistance genes

Antibiotic resistance surveillance consists of constant monitoring and inspection of resistant strains which enables early detection and efficient decision making to protect public health. Antibiotic Susceptibility Tests (AST) are a crucial part of antibiotic resistance surveillance where microbiology laboratories test which antibiotics would be effective against bacteria isolated from clinical samples (Johnson, 2015). This is especially critical when treating patients with serious conditions. However, one of the drawbacks of AST is they take microbiology facilities and expert personnel to derive useful information which might not be available everywhere. AST can also take weeks for slow growing organisms such as *Mycobacterium tuberculosis* (Boolchandani et al., 2019).

To speed up this process we ought to take advantage of modern computational approaches such as bioinformatics and Artificial Intelligence (AI) techniques. In Chapter 3, I propose a machine learning model with deep neural networks to predict the class of antibiotic resistance using protein sequences. This is specially significant when bacteria show secondary antibiotic resistance by gaining resistance genes from other bacteria.

1.4 Machine learning in computational biology

Machine learning is a sub-field of computer science where algorithms attempt prediction by learning from data. There are mainly two types of machine learning algorithm in practice - 1) supervised learning and 2) unsupervised learning. In supervised learning, we have a dataset where we also have the true answers, and the algorithm learns by trying to minimize its error comparing with the true answers. In contrast, unsupervised learning is usually needed when we have datasets where we do not know the true answers, and therefore, we are trying to capture some underlying structure of the data. Deep learning is a sub-field of machine learning where artificial neural networks with many hidden layers are used to learn from data, and predict on future unseen data (LeCun et al., 2015). Deep learning is currently revolutionizing the field of computational biology (Ching et al., 2018). It has been used in gaining biological insights from bacteria gene expression data (Tan et al., 2016), to predict sequence specificities of DNA binding proteins (Alipanahi et al., 2015), predict promoter sequences (Umarov and Solovyev, 2017), predict enhancers (Kelley et al., 2016), protein secondary structure and contact map prediction (Wang et al., 2016a,b), protein-protein interaction prediction (Du et al., 2017), protein family classification (Asgari and Mofrad, 2015), and in many other applications across different fields. In this dissertation, I am using deep learning along with the transfer learning technique to predict function from protein sequences. Transfer learning is helpful when we have a small dataset where we know the true answers but we also have a large dataset where true answers are not available. In the next section I go into transfer learning in more detail.

1.5 Transfer learning

Transfer learning is a sub-field of machine learning where knowledge extracted from solving a problem is used to solve a different problem more efficiently (Ruder et al., 2019). Formally, transfer learning involves a source domain $\mathcal{D}_S$ and related learning task $\mathcal{L}_S$. It also involves a target domain $\mathcal{D}_T$ and related learning task $\mathcal{L}_T$. Transfer learning's goal is to improve the learning task $\mathcal{L}_T$ using the knowledge extracted from solving the learning task $\mathcal{L}_S$ (Pan and Yang, 2009).

According to the taxonomy defined in Pan and Yang (2009), there are mainly two different types of transfer learning.

- Inductive transfer learning: in this setting, the learning task in the target domain is different from the learning task in the source domain. In this case, usually there is small labeled data for the target domain learning task. That is why the source domain learning task is used to boost the performance of the target domain learning task. An example would be training a deep neural network as a language model (source domain learning task), and then use the trained neural network to classify movie reviews as positive or negative (target domain learning task).

- Transductive transfer learning: in this setting, the learning task in the target domain and the source domain are same. But we do not have any labeled data for the target domain learning task. An example would be training a deep neural network to do parts of speech tagging for sentences in a language where lots of labeled data is available, and then use the trained model to do parts of speech tagging in another language where no labeled data is available.

In this dissertation, the transfer learning methods mainly involve inductive transfer learning. In Chapter 2, I use a shallow transfer learning method called word2vec to learn protein sequence representation. Machine learning models need some kind of numeric representation as input for the protein sequences. To acquire that representation, first, I use a word2vec learning algorithm to learn a numeric vector representation of each 3-mers (contiguous size 3 amino acid words) using lots of unlabeled protein sequences. Then I use these representations of 3-mers to represent both

bacteriocin and non-bacteriocin protein sequences and classify them. The transfer of 3-mer representations improves the performance of our final classification task significantly. Word2vec learns numeric representation of each 3-mer by taking into account its surrounding 3-mers in all protein sequences. This kind of contextual information helps capture meaningful relationship between different 3-mers which helps in distinguishing functionalities of individual protein sequences. Chapter 2 provides a detailed explanation of word2vec.

In Chapter 3, again, I use another transfer learning method based on language modeling to learn numeric representations of protein sequences. We have a small labeled dataset of protein sequences, labeled with their antibiotic resistance class. To boost the performance of the antibiotic classification task, I trained a deep neural network as a language model on unlabeled protein sequences, and learn numeric representations for them. Then, I used these representations to represent protein sequences in our small labeled dataset of antibiotic resistance classification, and try to classify. Language modeling learns the probability distribution over a sequence of amino acids by trying to predict the next amino acid given some amino acids of the sequence. This task helps the deep neural network learn numeric representation for individual protein sequences. Chapter 3 delves deep into how language models work, and how it can be used to learn protein sequence representations.

1.6 Reliable uncertainty estimation

When a neural network classification model tries to classify an instance which does not belong to any of the classes the model has been trained to predict, what does it do? Usually, the model still classifies that instance into one of the classes it was trained to predict with a high probability. This can be fatal in critical applications such as in self-driving cars. When a model receives an instance that does not belong to any of the classes it was trained to predict, ideally, it needs to relay that information by providing a low probability score. This kind of reliable uncertainty estimate is also crucial for antibiotic resistance class prediction. In Chapter 4, I used a novel optimization

algorithm to train a deep neural network which provides reliable uncertainty estimate faced with genes that do not confer antibiotic resistance or might not even be from bacteria.

1.7 Dissertation structure

This dissertation has 5 chapters. Chapter 1 provides an overview of the problems the dissertation is attempting to tackle, and the relevant machine learning techniques.

Chapter 2 is an accepted publication at *Bioinformatics* where I try to predict bacteriocins from protein sequences. I use a word2vec algorithm to learn representations for our protein sequences, and use these representations for our final classification task. I show that our transfer learning approach performs better than traditional alignment based approaches such as BLAST or HMMER.

Chapter 3 is a soon to be submitted manuscript where I use a deep learning model to predict antibiotic classes from protein sequences. Here, again, protein sequence representations are learnt from a large number of unlabeled protein sequences using a deep neural network as a language model. I use these learned representations in our final classification task, and show that this type of transfer learning significantly improves the model's performance.

Chapter 4 is also a soon to be submitted manuscript that introduces an orthogonal topic to the transfer learning approach. In this chapter, I aim to provide a method to gain reliable uncertainty estimates for antibiotic resistance gene classification when the model is tested on sequences that do not confer antibiotic resistance or might not be even from bacteria. Specifically, I applied a novel optimization algorithm to train our neural networks to gain better uncertainty estimate of our predictions. Finally, in Chapter 5, I summarize the contribution of this dissertation, and discuss future directions of my research.

1.8 References

Alipanahi, B., Delong, A., Weirauch, M. T., and Frey, B. J. (2015). Predicting the sequence specificities of dna-and rna-binding proteins by deep learning. *Nature biotechnology*, 33(8):831.

Asgari, E. and Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287.

Boolchandani, M., DSouza, A. W., and Dantas, G. (2019). Sequencing-based methods and resources to study antimicrobial resistance. *Nature Reviews Genetics*, page 1.

Ching, T., Himmelstein, D. S., Beaulieu-Jones, B. K., Kalinin, A. A., Do, B. T., Way, G. P., Ferrero, E., Agapow, P.-M., Zietz, M., Hoffman, M. M., et al. (2018). Opportunities and obstacles for deep learning in biology and medicine. *Journal of The Royal Society Interface*, 15(141):20170387.

Du, X., Sun, S., Hu, C., Yao, Y., Yan, Y., and Zhang, Y. (2017). Deepppi: boosting prediction of protein–protein interactions with deep neural networks. *Journal of chemical information and modeling*, 57(6):1499–1510.

for Disease Control, C. and Prevention (2018). About Antimicrobial Resistance. https://www.cdc.gov/drugresistance/about.html.

Johnson, A. P. (2015). Surveillance of antibiotic resistance. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 370(1670):20140080.

Kelley, D. R., Snoek, J., and Rinn, J. L. (2016). Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome research*, 26(7):990–999.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

Nicolaou, K. C. and Rigol, S. (2018). A brief history of antibiotics and select advances in their synthesis. *The Journal of antibiotics*, 71(2):153.

Pan, S. J. and Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

Piddock, L. J. (2012). The crisis of no new antibioticswhat is the way forward? *The Lancet infectious diseases*, 12(3):249–253.

Read, A. F. and Woods, R. J. (2014). Antibiotic resistance management. *Evolution, medicine, and public health*, 2014(1):147.

Ruder, S., Peters, M. E., Swayamdipta, S., and Wolf, T. (2019). Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18.

Sengupta, S., Chattopadhyay, M. K., and Grossart, H.-P. (2013). The multifaceted roles of antibiotics and antibiotic resistance in nature. *Frontiers in microbiology*, 4:47.

Shrestha, L. B. et al. (2005). Life expectancy in the united states. Congressional Information Service, Library of Congress.

Tan, J., Hammond, J. H., Hogan, D. A., and Greene, C. S. (2016). Adage-based integration of publicly available pseudomonas aeruginosa gene expression data with denoising autoencoders illuminates microbe-host interactions. *MSystems*, 1(1):e00025–15.

Umarov, R. K. and Solovyev, V. V. (2017). Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PloS one*, 12(2):e0171410.

Ventola, C. L. (2015). The antibiotic resistance crisis: part 1: causes and threats. *Pharmacy and therapeutics*, 40(4):277.

Wang, S., Peng, J., Ma, J., and Xu, J. (2016a). Protein secondary structure prediction using deep convolutional neural fields. *Scientific reports*, 6:18962.

Wang, S., Sun, S., and Xu, J. (2016b). Auc-maximized deep convolutional neural fields for protein sequence labeling. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 1–16. Springer.

Wright, G. D. (2014). Something old, something new: revisiting natural products in antibiotic drug discovery. *Canadian journal of microbiology*, 60(3):147–154.

CHAPTER 2. IDENTIFYING ANTIMICROBIAL PEPTIDES USING WORD EMBEDDING WITH DEEP RECURRENT NEURAL NETWORKS

Modified from a paper accepted by *Bioinformatics*

Md-Nafiz Hamid[1,2] and Iddo Friedberg[1,2]

[1]Program in Bioinformatics and Computational Biology, Iowa State University, Ames, Iowa, USA

[2]Veterinary Microbiology and Preventive Medicine, Iowa State University, Ames, Iowa, USA.

2.1 Abstract

Antibiotic resistance constitutes a major public health crisis, and finding new sources of antimicrobial drugs is crucial to solving it. Bacteriocins, which are bacterially-produced antimicrobial peptide products, are candidates for broadening the available choices of antimicrobials. However, the discovery of new bacteriocins by genomic mining is hampered by their sequences' low complexity and high variance, which frustrates sequence similarity-based searches.Here we use word embeddings of protein sequences to represent bacteriocins, and apply a word embedding method that accounts for amino acid order in protein sequences, to predict novel bacteriocins from protein sequences without using sequence similarity. Our method predicts, with a high probability, six yet unknown putative bacteriocins in *Lactobacillus*. Generalized, the representation of sequences with word embeddings preserving sequence order information can be applied to peptide and protein classification problems for which sequence similarity cannot be used. Data and source code for this project are freely available at: https://github.com/nafizh/Bacteriocin_paper

2.2 Introduction

The discovery of antibiotics ranks among the greatest achievements of modern medicine. Antibiotics have eradicated many infectious diseases and enabled many medical procedures that would

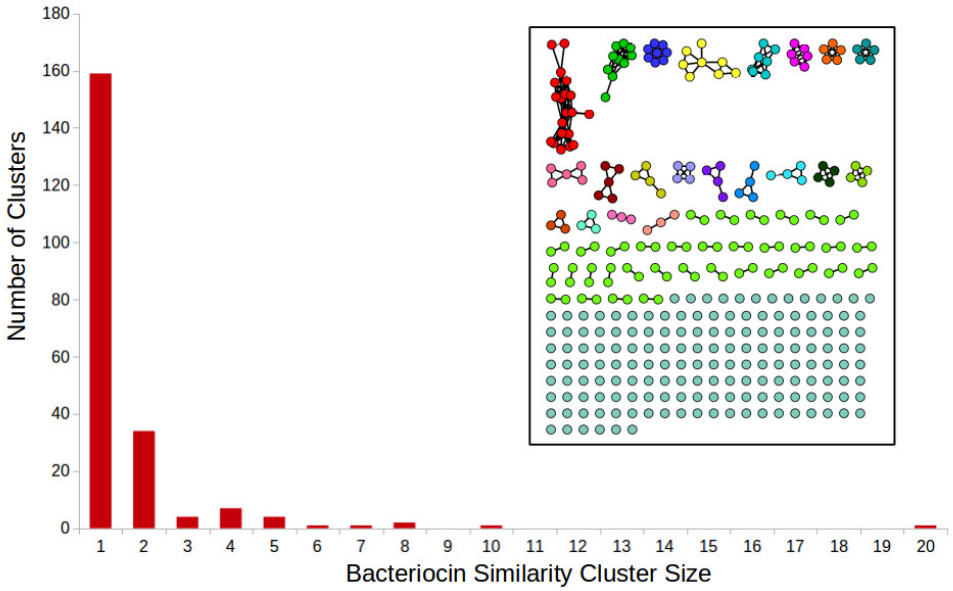


Figure 2.1: Inset: sequence similarity network for all of the bacteriocins present in the BAGEL dataset. Each node is a bacteriocin. There exists an edge between two nodes if the sequence identity between them is $\geq 35\%$ using pairwise all-*vs*-all BLAST. The bar chart shows cluster sizes.

have otherwise been fatal, including modern surgery, organ transplants, and immunosupressive treatments. However, due to the prevalent use of antibiotics in healthcare and agriculture, antibiotic resistant bacteria have been emerging in unprecedented scales. Each year, 23,000 people in the US alone die from infections caused by antibiotic resistant bacteria (for Disease Control and (US), 2013). One strategy to combat antibiotic resistance is to search for antimicrobial compounds other than antibiotics, and which may not be as prone to resistance. A promising class of such compounds are the peptide-based antimicrobials known as bacteriocins (Willey and van der Donk, 2007; Guder et al., 2000). Bacteriocins comprise a broad spectrum of bacterial ribosomal products, and with the increased sequencing of genomes and metagenomes, we are presented with a wealth of data that also include genes encoding bacteriocins. Bacteriocins generally have a narrow killing spectrum making them attractive antimicrobials that would generate less resistance (Riley and Wertz, 2002).

Several computational tools and databases have been developed to aid discovery and identification of bacteriocins. BAGEL (van Heel et al., 2013) is a database and a homology-based search tool that includes a large number of experimentally verified annotated bacteriocin sequences. BACTIBASE (Hammami et al., 2010) is a similar tool, which also contains predicted sequences. AntiSMASH (Weber et al., 2015) is a platform for genome mining for secondary metabolite producers, which also includes bacteriocin discovery. BOA (Bacteriocin Operon Associator) (Morton et al., 2015) identifies possible bacteriocins by searching for homologs of *context genes*: genes that are associated with the transport, immunity, regulation, and post-translational modification of bacteriocins. RiPPquest (Mohimani et al., 2014) is an automated mass spectrometry based method towards finding Ribosomally synthesized and posttransationally modified peptides (RiPPs) which may include bacteriocins. Recently, MetaRiPPquest (Mohimani et al., 2017) improved upon RiPPquest by using high-resolution mass spectrometry, and increasing the search space for RiPPs. However, bacteriocins are still hard to identify using standard bioinformatics methods. The challenge in detecting bacteriocins is twofold: first, a small number of positive examples of known bacteriocin sequences, and second, bacteriocins are highly diverse in sequence, and therefore challenging to discover using standard sequence-similarity based methods, (Figure 2.1).

To address these challenges we present a novel method to identify bacteriocins using word embedding. We represent protein sequences using Word2vec (Mikolov et al., 2013). Using this representation, we use a deep Recurrent Neural Network (RNN) to distinguish between bacteriocin and non-bacteriocin sequences. Our results show that a word embedding representation with RNNs can classify bacteriocins better than current tools and algorithms for biological sequence classification.

2.3 Methods

2.3.1 The Representation of Proteins with Word Embedding Vectors

Word embedding is a set of techniques in natural language processing in which words from a vocabulary are represented as vectors using a large corpus of text as the input. One word embedding technique is Word2vec, where similar vector representations are assigned to words that appear in similar contexts based on word proximity as gathered from a large corpus of documents. After training on a large corpus of text, the vectors representing many words show interesting and useful contextual properties. For example, after training on a large corpus of English language documents, given vectors representing words that are countries and capitals, $\overrightarrow{Madrid} - \overrightarrow{Spain} + \overrightarrow{France}$ will result in a vector that is similar to $\overrightarrow{Paris}$, more than other vectors in the corpus (Mikolov et al., 2013). This type of representation has led to better performance in downstream classification problems, including in biomedical literature classification (Minarro-Giménez et al., 2014; Chen et al., 2018), annotations (Zwierzyna and Overington, 2017; Duong et al., 2017), and genomic sequence classifications (Mejia Guerra and Buckler, 2017; Dutta et al., 2018; Zhang et al., 2018; Du et al., 2018).

The training for generating the vectors can be done in two ways: the continuous bag of words (CboW) model, or the skip-gram model (Mikolov et al., 2013). We adapted Word2vec for protein representation as in (Asgari and Mofrad, 2015), using the skip-gram model. Instead of the common representation of protein sequences as a collection of counts of n-grams (also known as k-mers) using a 20 letter alphabet, we represent protein sequences using embeddings for each n-gram, covering all possible amino-acid n-grams (we used $n = 3$, leading to $20^3 = 8,000$ trigrams). Each trigram is a "word", and the 8,000 words constitute the vocabulary. The Uniprot/TrEMBL database (Apweiler et al., 2004) constitutes the equivalent of the document corpus.

The skip-gram model is a neural network where the inputs and outputs of the network are one-hot vectors with our training instance input word and output word. A one-hot vector is a boolean

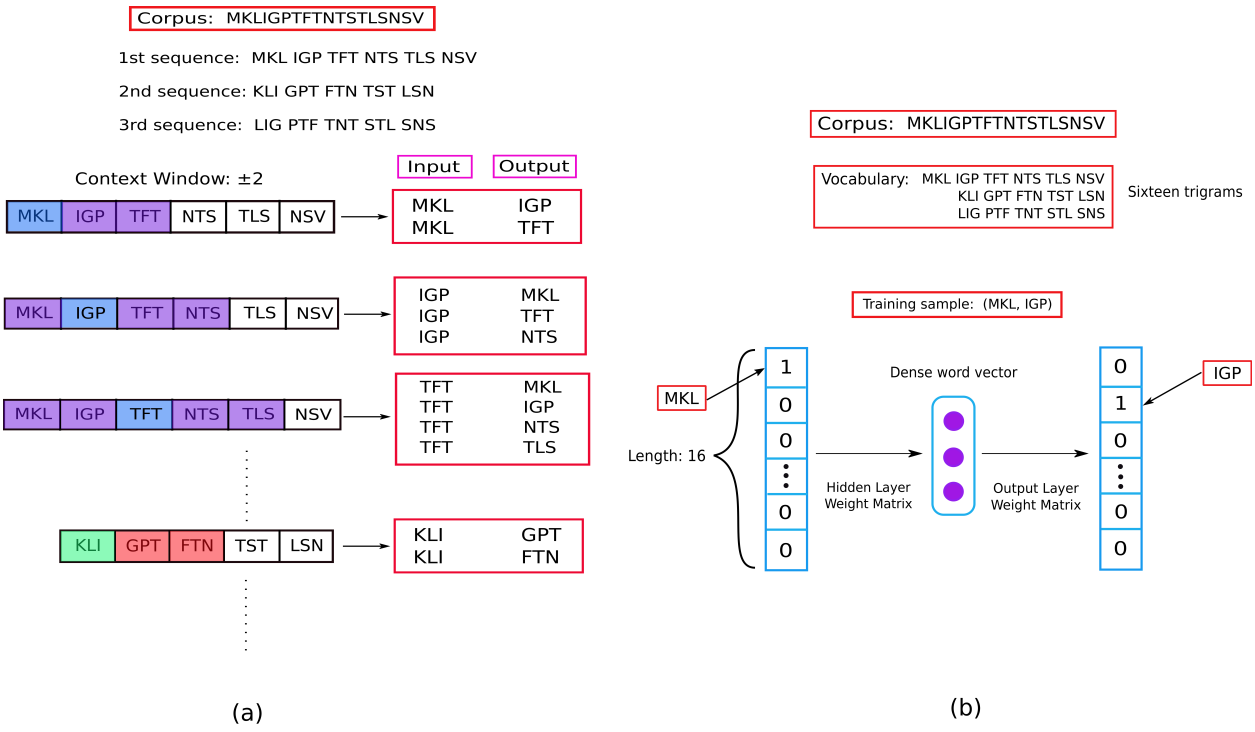


Figure 2.2: A simplified example showing representation learning for trigrams with skip-gram training. For simplicity, in the example, the vocabulary comprises of 16 words, the context window is ±2 (in our study, the vocabulary size was 8,000, and the context window ±5). (a) For each sequence in the TrEMBL database we created 3 sequences by starting the sequence from the first, second, and third amino acid as in (Asgari and Mofrad, 2015). This makes sure that we consider all of the overlapping trigrams for a protein sequence. A protein sequence, is then broken into trigrams, and training instances (input, output) are generated according to the size of the context window for the subsequent step of training a neural network. (b) The neural network architecture for training on all of the instances generated at (a). The diagram shows training on the instance where MKL is input, and IGP is output which is the first instance generated at (a). At the end of the training, for each trigram a dense word vector of size 200 is produced (center, purple circles).

vector of the size of the vocabulary (8,000 in our case, six in Figure 2.2), in which only the entry corresponding to the word of choice has a value of True. We generated the training instances using a context window of size ±5, where we took a word as input (in this case, a word is a trigram), and used the surrounding words within the context window as outputs. The process is explained in Figure 2.2. At the end of the training, a 200 dimensional vector for each trigram was generated by the neural network. The goal of this training was to have the 200 dimensional vectors capture information about the surroundings of each trigram that they are representing. In this fashion, we capture the contextual information for each trigram in our corpus of protein sequences. The size of the vector is a hyper-parameter which we decided upon based on the final supervised classification performance. Vectors of sizes 100, 200, and 300 were generated, and size 200 was chosen. Similarly, context window sizes of 3, 5 and 7 were tested, and size 5 was chosen.

2.3.2 Word2vec with a Recurrent Neural Network

Taking the word-embedding representation for each trigram present in a protein sequence, we used a Recurrent Neural Network (RNN) to take all trigram embedding vectors as its input to represent a protein sequence (Figure 2.3). Since RNNs share the same weights for all inputs in a temporal sequence, we took advantage of this architecture by using an embedding vector of size 200 for each overlapping trigram in a protein sequence. By using the embedding vectors of overlapping trigrams as temporal inputs to an RNN, we preserved the order of the trigrams in the protein sequence. Regarding the architecture of the RNN, we used a two-layer Bidirectional RNN with Gated Recurrent Units (GRU) to train on our data. Our hyper-parameters of number of neurons, network depth, and dropout (Srivastava et al., 2014b) rate were determined with nested cross-validation. Since we had a small dataset, we used a dropout rate of 0.5 for the first layer, and 0.7 for the second layer. Both layers had 32 GRU units. We used a fixed number of 100 epochs for training which was also decided by nested cross-validation. For optimization, the Adam (Kingma and Ba, 2014) method was used.

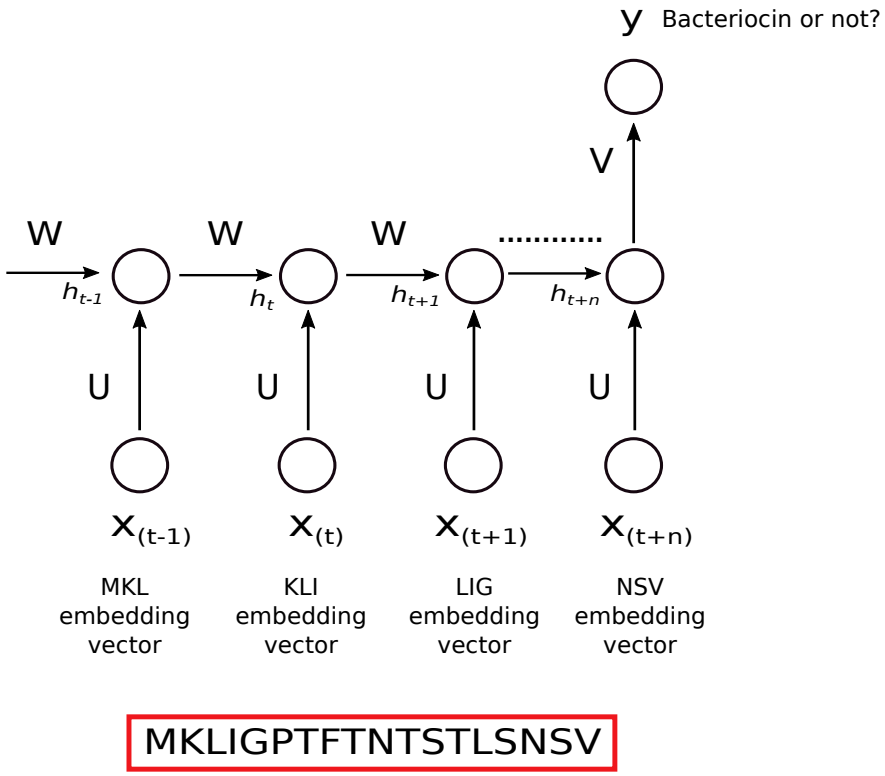


Figure 2.3: We used embedding vectors of each individual overlapping trigram present in a protein sequence as input into a Recurrent Neural Network. $X_{(t)}$ is the input at time step t. In our case, at each time step t, input is the embedding vector of the trigram at that time step. $h_{(t)}$ is the hidden state at time step t. It contains information from the previous inputs as well as the current input. This works like the memory of the network, and because of its mechanism, modern RNNs can preserve information over long ranges unlike traditional models like hidden Markov models. U, V, W are weights of the network. As they are being shared over all the inputs, this greatly reduces the number of parameters of the network helping towards generalization. At the end of the sequence, the network produces a prediction y of whether the sequence is a bacteriocin or not. In practice, we used a bidirectional RNN (not shown in figure).

2.3.3 Comparing with baseline methods

We compared the performance of our method with four baseline methods: (1) a simple trigram representation, (2) an averaged word-embedding representation, (3) BLAST (Altschul et al., 1997), and (4) HMMER3 (Eddy, 2011).

We used a trigram representation of sequences in bioinformatics to understand the gain of accuracy, if any, of using word embedding over simple trigram based representation. To implement the simple trigram representation, we created an 8,000 size vector for each sequence where the indices had counts for each occurrence of a trigram in that sequence. In this representation, the order of the trigrams is not preserved. Since the vector is sparse, we used truncated Singular Value Decomposition (SVD) to acquire the most importance features, and reduce the size of the vector (Figure 2.4). We tried vector sizes of 100 and 200, and used 200 as it led to better classification performance. We then used these vectors with a support vector machine (SVM), logistic regression (LR), decision tree (DT), and random forest (RF), to classify genes into bacteriocins and non-bacteriocins.

We also used an averaged word-embedding representation, and evaluated its performance with SVM, LR, DT, and RF. We summed the embedding vectors for each overlapping trigram in a sequence, and divided the sum by the length of the sequence. We then used this new mean embedding vector that is representative of the whole protein sequence with supervised learning algorithms.

We compared the performance of our method with BLAST, the method of choice for sequence similarity search and, by proxy, determination of gene and protein function. We use BLAST to see if machine learning based, alignment free methods do indeed improve performance over alignment based methods to identify potential bacteriocins. We used a 35% sequence identity score as a threshold to assign a bacteriocin label to a protein sequence. This threshold was used to increase BLAST's recall even at the expense of precision, and was based on the finding that 35% ID is the "Twilight Zone" of protein sequence alignments below which one cannot unambiguously distinguish between true and false sequence alignment, using protein structure as a standard (Rost, 1999).

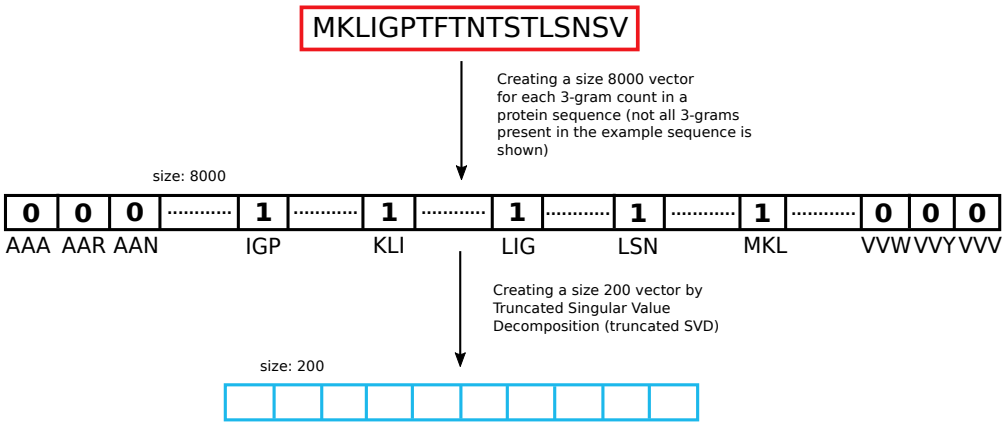


Figure 2.4: We represented each protein sequence with the overlapping trigram counts present in that sequence. This leads to a size 8,000 sparse vector. The vector was reduced to a vector of size 200 using Singular Value Decomposition. We used the size 200 vector as the baseline representation.

We also compared our performance with another popular alignment based method, HMMER3, which constructs profile hidden Markov models or pHMMs from multiple sequence alignments. In turn, the pHMMs serve as an accurate tool for sequence searching. Here we used bacteriocin pHMMs which we constructed using BOA (Morton et al., 2015). BOA uses the BAGEL (van Heel et al., 2013) dataset, and its homologs (BLAST e-value $< 10^{-5}$) against GenBank (Benson et al., 2014) bacterial database to build bacteriocin-specific pHMMs. We used the HMMSearch functionality provided by HMMER3, and use the pHMMs from BOA to measure performance against our test set in terms of precision, recall, and F_1 score.

2.3.4 Building the training dataset

We used 346 experimentally determined bacteriocin sequences of lengths $\geq$ 30aa from the BAGEL database as our positive bacteriocin training samples. For the negative training set, we used sequences from the Uniprot-Swissprot (Boutet et al., 2007) database. We took all the bacterial protein sequences from this database and used CD-HIT (Fu et al., 2012) with a 50% identity threshold to reduce redundancy. Then, for the primary negative training set, we took 346 sequences that had the keywords 'not anti-microbial', 'not antibiotic', 'not in plasmid', and that had the same length distribution as our positive bacteriocin sequences. We also generated two additional

negative datasets following the same steps as above, with no overlap in the sequences between the three sets. Because identical length sequences were already exhausted by the first negative set, the length distribution of the second and third negative sets are somewhat different than the positive bacteriocin set. Figure 2.5 shows the length distribution of the positive, and all three negative datasets.

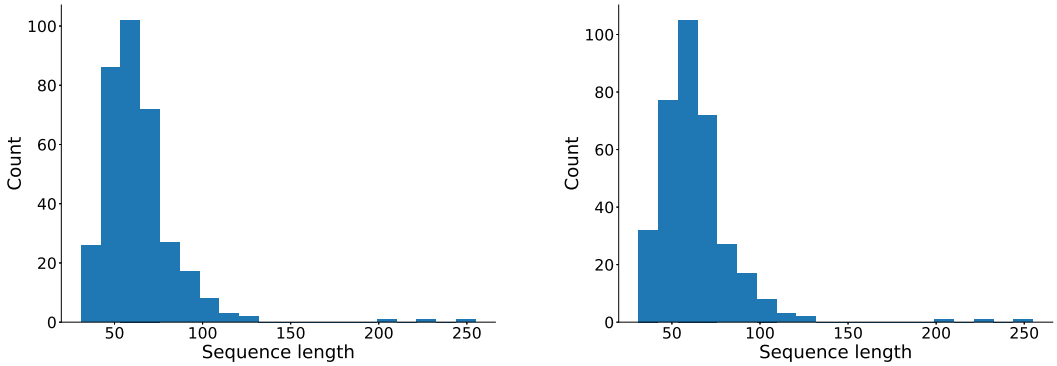
2.3.5 Identifying genomic regions for novel putative bacteriocins

To search for genomic regions with a higher probability of containing novel bacteriocins, we took advantage of the known proximity of *context genes* whose products assist in the transport, modification, and regulation of bacteriocins. Many bacteriocins have some or all of four types of context genes in proximity (de Vos et al., 1995; McAuliffe et al., 2001), (Figure 2.6). Having an experimentally verified set of fifty-four context genes from (Morton et al., 2015), we now aimed to expand it. To do so, we collected the annotation keywords for these context genes from the Refseq database,

We ran BLAST using all 1294 (54 experimentally verified and 1240 newly found) putative context genes against the whole bacteria RefSeq database (Pruitt et al., 2006) and we collected hits with an e-value $\leq 10^{-6}$. We separated all the hits by organism, arranged them by co-ordinates, and identified 50kb regions in the whole genome that have contiguous hits. We then ran our method on the ORFs that were not identified as conext genes, to see if we could identify new bacteriocin genes within these regions.

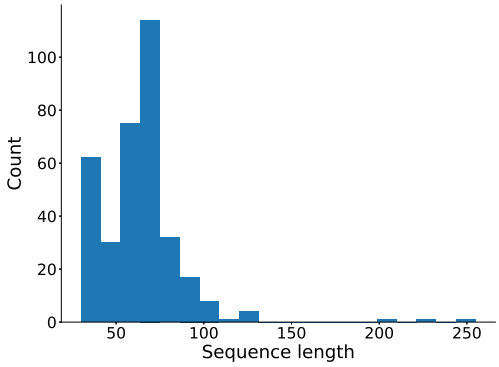
We used the following software tools in this study: Keras (Chollet et al., 2015), Scikit-learn (Pedregosa et al., 2011), Gensim (Řehůřek and Sojka, 2010), Matplotlib (Hunter, 2007), Jupyter notebooks (Kluyver et al., 2016), Biopython (Cock et al., 2009), Numpy, and Scipy (Walt et al., 2011).

We then took all the genes with similar keywords to our experimentally verified context gene set surrounding the BAGEL bacteriocins within a region of ±25kb. After running CD-HIT (Li and Godzik, 2006a) to remove redundancy, we had 1,240 new putative context genes.

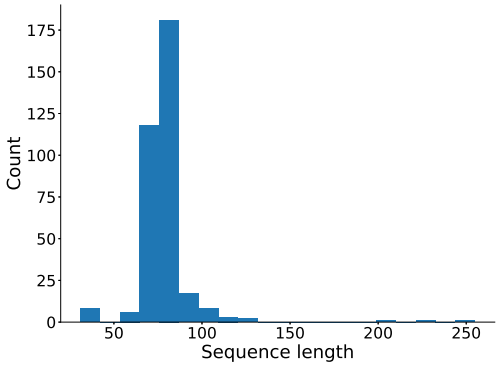


(a) positive set

(b) primary negative set



(a) 2nd negative set



(b) 3rd negative set

Figure 2.5: Sequence length distributions for the positive bacteriocin set, primary negative set, 2nd, and 3rd negative sets respectively. Mean lengths of training sets were 63.57aa (with the primary negative set), 63.53 (with second negative set) and 70.92 (with third negative set). See text for details.

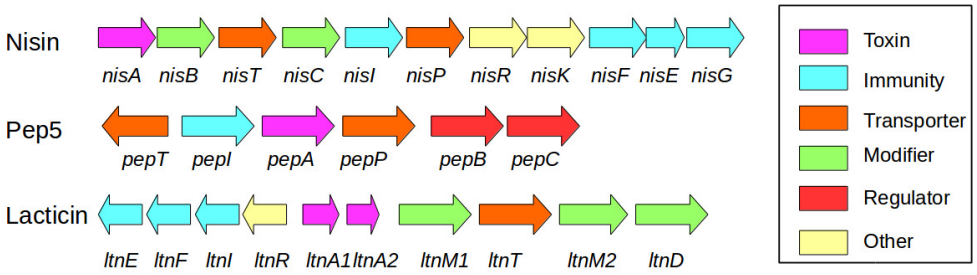


Figure 2.6: Bacteriocins with context genes. After de Vos et al. (1995).

2.3.6 Datasets

We performed 10× cross-validations on the three datasets we built where the datasets consist of positive bacteriocins from BAGEL, and the three negative datasets we built from Uniprot Swissprot database.

The cross-validation itself was done 50 times with different random seeds for all cases except for the RNN, BLAST, and HMMER for which it was done 10 times due to computational time demand. For BLAST, a 35% sequence identity score was used as a threshold for calling a result positive. We used the same cross-validation folds for BLAST as other algorithms where we BLASTed the test set against the training set. For HMMER, an e-value of $< 10^{-3}$ was used as the threshold for deciding if a sequence is bacteriocin. The reported results are the mean of 10× nested cross-validation done 50 times (10 times for RNN, BLAST and HMMER), and the standard error is from those 50 (10 for RNN, BLAST and HMMER) mean values.

2.4 Results

Table 2.1 and Figure 2.7 show a comparison of Word2vec, trigram representation, BLAST, and HMMER for predicting bacteriocins using the primary bacteriocin dataset in terms of precision, recall, and F_1 score.

Precision (Pr) Recall (Rc) and F_1 are defined as:

$$Pr = \frac{TP}{TP + FP};\ Rc = \frac{TP}{TP + FN};\ F_1 = 2 \times \frac{Pr \times Rc}{Pr + Rc}$$

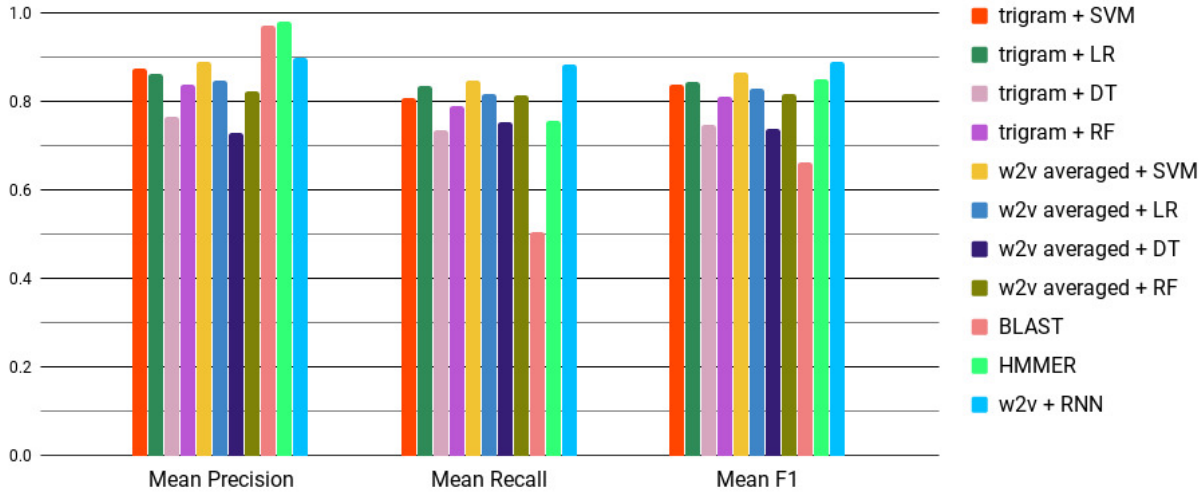


Figure 2.7: Mean F_1 scores of different algorithms with both Word2vec (w2v) and baseline representations. Error bars (using standard error) are too small to be shown. W2v + RNN (blue) provides the best F_1 score. See Table S1 for mean and standard errors values.

Where TP: True Positives, FP: False Positives, FN: False Negatives.

W2v+RNN provides the best recall, and F_1 score. HMMER and BLAST have better precision scores, which is expected as they only predict true positives by e-value and sequence identity respectively but they have high false negative rate. Using a simple trigram representation,SVM and LR perform similarly but with lower precision, recall, and F_1 score than w2v+RNN. Using mean Word2vec representation as input, LR, DT, and RF provides similar or worse F_1 score than the performances of those supervised methods with a simple trigram representation. Only mean Word2vec representation as input to an SVM shows a competitive performance against W2v+RNN. Still, the difference in F_1 between mean w2v+SVM and Word2vec+RNN was statistically significant and shows that Word2vec+RNN performs better (one sided t-test, $p = 5.48 \times 10^{-8}$)

Figure 2.8 shows the precision-recall curves for w2v+RNN, averaged Word2vec representation with SVM, LR, RF. Also, simple trigram representation with SVM (trigram+SVM), trigram+LR, trigram+RF, and BLAST. RNN has the largest area under the curve. W2v+SVM is competitive

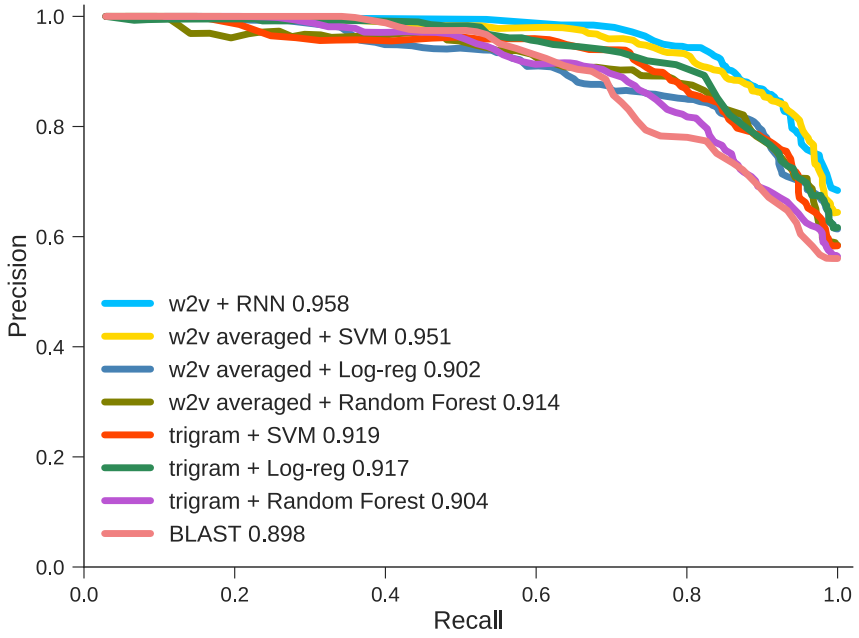


Figure 2.8: Mean precision-recall curves of one run of 10× cross validation for Word2vec with RNN, Support Vector Machine (SVM), Logistic Regression (Log-reg), Random Forest, and BLAST. Number in legend is area under the curve. w2v+RNN performs better than all the other methods.

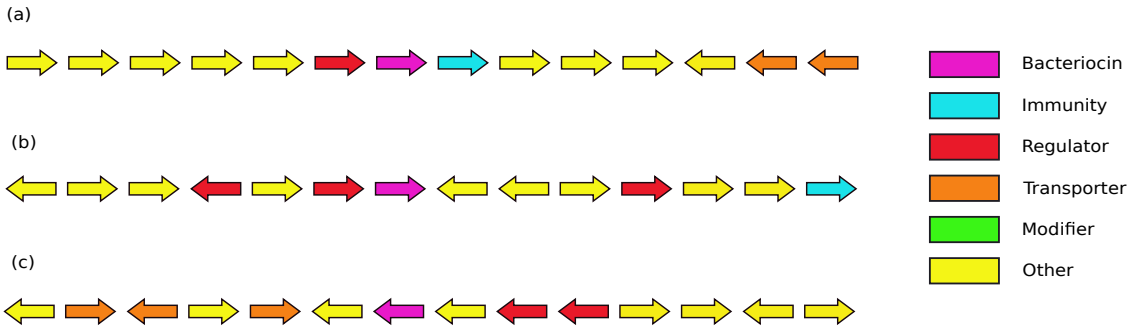


Figure 2.9: Context genes found surrounding the predicted bacteriocins within ±25kb range. (a) *Lactobacillus acidophilus NCFM* (Locus: NC_006814, putative bacteriocin: YP_193019.1, immunity: YP_193020.1, regulator: YP_193018.1, transporters: YP_193025.1, YP_193026.1) (b) *Lactobacillus helveticus R0052* (GenBnk: NC_018528, putative bacteriocin: YP_006656667.1, immunity: YP_006656674.1, regulator: YP_006656666.1, YP_006656664.1, YP_006656671.1) (c) *Lactobacillus helveticus CNRZ32* (GenBank ID: NC_021744, putative bacteriocin: YP_008236084.1, regulators: YP_008236086.1, YP_008236087.1, transporters: YP_008236082.1, YP_008236080.1, YP_008236079.1)

with w2v+RNN as was also seen in Figure 2.7. The curve for HMMER could not be shown as we need a confidence value for each prediction which HMMER does not provide.

Table 2.2 and 2.3 show the performance differences using the two other training data sets. The length distribution of the protein sequences in the positive and negative sets are different as mentioned in Methods. Looking at Table 2.2, the improvement in the w2v+RNN and the trigram based methods is evident, as well as the precision of HMMER. We assume the length disparity between the positive and negative sequences have helped in correctly classifying bacteriocins. Surprisingly, the precision of BLAST has decreased compared with its precision in the primary bacteriocin dataset. The performance of HMMER has largely remained the same over the different negative sets, with its predictions remaining more or less the same because of its low false positive rate. Table 2.3 shows the performance comparison for the third bacteriocin dataset. The length disparity between positive and negative sequences for the third dataset is even greater than the second bacteriocin dataset. SVM, LR, DT, and RF have all improved performance. SVM's precision is comparable to that of w2v+RNN. RNN still has the best recall and F_1 score. In contrast, BLAST's performance has significantly decreased indicating that somehow the length disparity is causing problems

in identifying true bacteriocins. Just like the second bacteriocin dataset, HMMER's performance remains almost the same with a slight improvement on the precision score.

After evaluating all the methods, we trained the best performing method, w2v+RNN on the whole dataset with the same hyper-parameters, and this final trained RNN was used to find new bacteriocins in the 50kb genomic regions that are that were identified based on context genes and are suspected of containing bacteriocin genes.

2.4.1 Results on 50kb Chromosomal Stretches

We applied our trained w2v+RNN model on the sequences identified from the 50kb regions (see Methods) to predict putative bacteriocins. The w2v+RNN model predicted 119 putative bacteriocins with a probability of ≥ 0.99. Figure 2.9 shows three of our predicted bacteriocins in their genomic neighborhood in *Lactobacillus*. We found several context genes surrounding these predicted bacteriocins, supporting our hypothesis that these bacteriocin predictions are valid.

2.5 Discussion

We developed a machine learning approach for predicting bacteriocins, a group of bacterial toxins which are challenging to discover using sequence similarity as they are, in many cases, non-homologous. Our approach does not require sequence similarity searches, and has discovered several putative bacteriocins with a high probability. The Word2vec representation takes advantage of the large volume of unlabeled bacterial protein sequences available, and can be used in other machine learning tasks in computational biology to represent protein sequences for discovering functional similarities that cannot be discovered from sequence similarity. We used the embedding vectors for each overlapping trigram in a protein sequence, and used them as input in a temporal order for an RNN, and with heavy regularization, it performed well. We hypothesize that the unsupervised step helped transfer important information through the trigram vectors such that the RNN's task was made easier. Another reason we used RNN, is that we can represent each protein sequence as overlapping trigrams. As a result, vectors of size 200 representing each subsequent trigram can

be fed into the RNN without dramatically increasing the feature space as opposed to SVM or Logistic Regression. For SVM or LR we would have needed to find another way to represent a protein sequence from its overlapping trigram vectors, so that its feature size does not overpower the number of training sequences available to us. For example, in the work by *Asgari et al.* (2015), the researchers summed up all the overlapping trigram vectors to represent a protein sequence. In this paper, we averaged the overlapping trigram vectors to represent a protein sequence, and used that as a baseline. We also built three different datasets using different sets of negative bacteriocin examples. All the methods except w2v+RNN and averaged w2v+SVM struggled to identify true bacteriocins in the primary bacteriocin dataset where the length distribution for positive and negative bacteriocins is exact. This is also the reason we used the primary bacteriocin dataset as the final dataset to train our RNN model before applying it to find novel bacteriocins in *Lactobacillus*. Compared with the primary bacteriocin dataset, the other methods except BLAST and HMMER have had improved performance as the differences in length distribution of positive and negative sequences increased in the second and third bacteriocin dataset. The BOA study (Morton et al., 2015) supplied us with pHMMs that were built using many sequences including the BAGEL dataset, and used with HMMER. Yet tested against the BAGEL sequences, HMMER's precision is high but the recall remained low compared with w2v+RNN.

Despite the training set being small, with proper regularization our RNN model provides a better precision than all the other methods except BLAST and HMMER, and better recall than all other methods. We argue that word embedding and RNN can be used to boost the prediction powers of machine learning models in sequence-based classification problems in biology. Our models also provide us with an associated confidence score, which is useful for experimentalists who wish to apply this method towards genome mining. We chose a threshold of 0.99 for RNN to provide the list of putative predictions. Although our training set is balanced in terms of bacteriocins and non-bacteriocins, the number of bacteriocin sequences in the microbial sequence universe is much lower. Finally, we provide six protein sequences that our model predicted to be bacteriocins, with a probability of ≥ 0.99, where we could also find putative context genes. We also provide a set

of total 119 sequences predicted by w2v+RNN with a probability of greater than 0.99. None of these sequences could be detected against known bacteriocins when we used BLAST against the *nr* database with an e-value $\leq 10^{-3}$.

Historically, the use of bioinformatics prediction methods has favored high precision over high recall, as a large number of false positive findings can be costly for experiments that verify predictions. However, there are cases where a high recall method is appropriate. For example, with the need to cast a wider net in identifying potential drug candidates, driven by decrease drug scanning costs. By employing a high recall method and choosing an appropriate accuracy threshold, experimentalists can calibrate the precision / recall trade off needed to optimize the functional testing of novel peptides.

Protein classification tasks are typically based on some form of sequence similarity as an indicator for evolutionary relatedness. However, in many cases non-orthologous replacements occur, where two non-homologous proteins perform the same function. Non-orthologous function replacements have been detected using natural language processing (Verspoor et al., 2012), genomic context methods (Overbeek et al., 1999; Huynen et al., 2000; Enault et al., 2005), and other combined methods (Franceschini et al., 2013). However, such methods require associated metadata or contextual genomic information. Here we present a solution to find functionally similar non-orthologs that does not require gathering these metadata, but does require a dataset of positive and negative examples. We therefore recommend that word embedding be explored for function classification involving dissimilar biological sequences.

2.6 Appendix: supplementary material

Table 2.1: Primary Bacteriocin dataset. Comparison between word2vec and trigram representation. SVM:Support Vector Machine; LogReg: logistic regression; DT: decision tree; RF: random forest, w2v + RNN: Recurrent Neural Network with word2vec representation.

	Mean Precision	**Mean Recall**	**Mean F_1**
trigram+SVM	0.875 ± 0.001	0.808 ± 0.002	0.838 ± 0.001
trigram+LogReg	0.864 ± 0.002	0.837 ± 0.002	0.846 ± 0.001
trigram+DT	0.767 ± 0.002	0.735 ± 0.002	0.747 ± 0.001
trigram+RF	0.838 ± 0.001	0.791 ± 0.001	0.812 ± 0.001
w2v averaged+SVM	0.889 ± 0.0007	0.848 ± 0.001	0.867 ± 0.0008
w2v averaged+LogReg	0.848 ± 0.001	0.817 ± 0.001	0.831 ± 0.0009
w2v averaged+DT	0.825 ± 0.001	0.813 ± 0.002	0.738 ± 0.002
w2v averaged+DRF	0.838 ± 0.001	0.791 ± 0.001	0.817 ± 0.001
BLAST	0.972 ± 0.0006	0.506 ± 0.002	0.663 ± 0.002
HMMER	**0.981 ± 0.0002**	0.757 ± 0.0001	0.852 ± 0.0003
w2v + RNN	0.898 ± 0.003	**0.883 ± 0.003**	**0.889 ± 0.001**

Table 2.2: Second Bacteriocin dataset.

	Mean Precision	**Mean Recall**	**Mean F_1**
SVM	0.902 ± 0.001	0.835 ± 0.001	0.865 ± 0.0009
LogReg	0.891 ± 0.001	0.871 ± 0.001	0.878 ± 0.001
DT	0.806 ± 0.002	0.767 ± 0.002	0.782 ± 0.001
RF	0.858 ± 0.001	0.792 ± 0.001	0.822 ± 0.001
BLAST	0.909 ± 0.002	0.504 ± 0.002	0.645 ± 0.001
HMMER	**0.985 ± 0.0001**	0.757 ± 0.0001	0.854 ± 0.0003
w2v + RNN	0.924 ± 0.002	**0.898 ± 0.001**	**0.909 ± 0.001**

Table 2.3: Third Bacteriocin dataset.

	Mean Precision	**Mean Recall**	**Mean F_1**
SVM	0.938 ± 0.001	0.898 ± 0.001	0.916 ± 0.0009
LogReg	0.916 ± 0.001	0.891 ± 0.001	0.902 ± 0.0007
DT	0.887 ± 0.001	0.856 ± 0.001	0.869 ± 0.001
RF	0.889 ± 0.001	0.878 ± 0.001	0.882 ± 0.001
BLAST	0.747 ± 0.004	0.504 ± 0.002	0.599 ± 0.002
HMMER	**0.992 ± 0.0001**	0.757 ± 0.0001	0.857 ± 0.0003
w2v + RNN	0.937 ± 0.002	**0.921 ± 0.002**	**0.928 ± 0.002**

2.7 References

Altschul, S. F., Madden, T. L., Schäffer, A. A., Zhang, J., Zhang, Z., Miller, W., and Lipman, D. J. (1997). Gapped blast and psi-blast: a new generation of protein database search programs. *Nucleic acids research*, 25(17):3389–3402.

Apweiler, R., Bairoch, A., Wu, C. H., Barker, W. C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., et al. (2004). Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl 1):D115–D119.

Asgari, E. and Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287.

Benson, D. A., Clark, K., Karsch-Mizrachi, I., Lipman, D. J., Ostell, J., and Sayers, E. W. (2014). Genbank. *Nucleic Acids Research*, 42(D1):D32–D37.

Boutet, E., Lieberherr, D., Tognolli, M., Schneider, M., and Bairoch, A. (2007). Uniprotkb/swiss-prot: the manually annotated section of the uniprot knowledgebase. *Plant bioinformatics: methods and protocols*, pages 89–112.

Chen, Z., He, Z., Liu, X., and Bian, J. (2018). Evaluating semantic relations in neural word embeddings with biomedical and general domain knowledge bases. *BMC medical informatics and decision making*, 18(Suppl 2).

Chollet, F. et al. (2015). Keras.

Cock, P. J., Antao, T., Chang, J. T., Chapman, B. A., Cox, C. J., Dalke, A., Friedberg, I., Hamelryck, T., Kauff, F., Wilczynski, B., and de Hoon, M. J. (2009). Biopython: freely available python tools for computational molecular biology and bioinformatics. *Bioinformatics (Oxford, England)*, 25(11):1422–1423.

de Vos, W. M., Kuipers, O. P., van der Meer, J. R., and Siezen, R. J. (1995). Maturation pathway of nisin and other lantibiotics: post-translationally modified antimicrobial peptides exported by gram-positive bacteria. *Molecular microbiology*, 17(3):427–437.

Du, J., Jia, P., Dai, Y., Tao, C., Zhao, Z., and Zhi, D. (2018). Gene2vec: Distributed representation of genes based on co-expression. *bioRxiv*.

Duong, D., Eskin, E., and Li, J. (2017). A novel word2vec based tool to estimate semantic similarity of genes by using gene ontology terms. *bioRxiv*.

Dutta, A., Dubey, T., Singh, K. K. K., and Anand, A. (2018). Splicevec: Distributed feature representations for splice junction prediction. *Computational biology and chemistry*, 74:434–441.

Eddy, S. R. (2011). Accelerated profile hmm searches. *PLOS Computational Biology*, 7(10):e1002195+.

Enault, F., Suhre, K., and Claverie, J.-M. M. (2005). Phydbac "gene function predictor": a gene annotation tool based on genomic context analysis. *BMC bioinformatics*, 6(1):247+.

for Disease Control, C. and (US), P. (2013). *Antibiotic resistance threats in the United States, 2013*. Centers for Disease Control and Prevention, US Department of Health and Human Services.

Franceschini, A., Szklarczyk, D., Frankild, S., Kuhn, M., Simonovic, M., Roth, A., Lin, J., Minguez, P., Bork, P., von Mering, C., and Jensen, L. J. (2013). String v9.1: protein-protein interaction networks, with increased coverage and integration. *Nucleic acids research*, 41(Database issue):D808–D815.

Fu, L., Niu, B., Zhu, Z., Wu, S., and Li, W. (2012). Cd-hit: accelerated for clustering the next-generation sequencing data. *Bioinformatics*, 28(23):3150–3152.

Guder, A., Wiedemann, I., and Sahl, H.-G. (2000). Posttranslationally modified bacteriocinsthe lantibiotics. *Biopolymers*, 55(1):62–73.

Hammami, R., Zouhir, A., Le Lay, C., Hamida, J. B., and Fliss, I. (2010). Bactibase second release: a database and tool platform for bacteriocin characterization. *Bmc Microbiology*, 10(1):22.

Hunter, J. D. (2007). Matplotlib: A 2d graphics environment. *Computing In Science & Engineering*, 9(3):90–95.

Huynen, M., Snel, B., Lathe, W., and Bork, P. (2000). Predicting protein function by genomic context: quantitative evaluation and qualitative inferences. *Genome research*, 10(8):1204–1210.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B. E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J. B., Grout, J., Corlay, S., et al. (2016). Jupyter notebooks-a publishing format for reproducible computational workflows. In *ELPUB*, pages 87–90.

Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics (Oxford, England)*, 22(13):1658–1659.

McAuliffe, O., Ross, R. P., and Hill, C. (2001). Lantibiotics: structure, biosynthesis and mode of action. *FEMS microbiology reviews*, 25(3):285–308.

Mejia Guerra, M. K. and Buckler, E. S. (2017). k-mer grammar uncovers maize regulatory architecture. *bioRxiv*.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Minarro-Giménez, J. A. A., Marín-Alonso, O., and Samwald, M. (2014). Exploring the application of deep learning techniques on medical text corpora. *Studies in health technology and informatics*, 205:584–588.

Mohimani, H., Gurevich, A., Alexander, K. L., Naman, C. B., Leao, T., Glukhov, E., Moss, N. A., Knaan, T. L., Vargas, F., Nothias, L.-F., et al. (2017). Metarippquest: A peptidogenomics approach for the discovery of ribosomally synthesized and post-translationally modified peptides. *bioRxiv*, page 227504.

Mohimani, H., Kersten, R. D., Liu, W.-T., Wang, M., Purvine, S. O., Wu, S., Brewer, H. M., Pasa-Tolic, L., Bandeira, N., Moore, B. S., et al. (2014). Automated genome mining of ribosomal peptide natural products. *ACS chemical biology*, 9(7):1545–1551.

Morton, J. T., Freed, S. D., Lee, S. W., and Friedberg, I. (2015). A large scale prediction of bacteriocin gene blocks suggests a wide functional spectrum for bacteriocins. *BMC bioinformatics*, 16(1):381.

Overbeek, R., Fonstein, M., D'Souza, M., Pusch, G. D., and Maltsev, N. (1999). The use of gene clusters to infer functional coupling. *Proceedings of the National Academy of Sciences of the United States of America*, 96(6):2896–2901.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Pruitt, K. D., Tatusova, T., and Maglott, D. R. (2006). Ncbi reference sequences (refseq): a curated non-redundant sequence database of genomes, transcripts and proteins. *Nucleic acids research*, 35(suppl_1):D61–D65.

Řehůřek, R. and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA.

Riley, M. A. and Wertz, J. E. (2002). Bacteriocins: evolution, ecology, and application. *Annual Reviews in Microbiology*, 56(1):117–137.

Rost, B. (1999). Twilight zone of protein sequence alignments. *Protein engineering*, 12(2):85–94.

Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

van Heel, A. J., de Jong, A., Montalban-Lopez, M., Kok, J., and Kuipers, O. P. (2013). Bagel3: automated identification of genes encoding bacteriocins and (non-) bactericidal posttranslationally modified peptides. *Nucleic acids research*, 41(W1):W448–W453.

Verspoor, K. M., Cohn, J. D., Ravikumar, K. E., and Wall, M. E. (2012). Text mining improves prediction of protein functional sites. *PloS one*, 7(2):e32171+.

Walt, S. v. d., Colbert, S. C., and Varoquaux, G. (2011). The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22–30.

Weber, T., Blin, K., Duddela, S., Krug, D., Kim, H. U., Bruccoleri, R., Lee, S. Y., Fischbach, M. A., Müller, R., Wohlleben, W., et al. (2015). antismash 3.0a comprehensive resource for the genome mining of biosynthetic gene clusters. *Nucleic acids research*, 43(W1):W237–W243.

Willey, J. M. and van der Donk, W. A. (2007). Lantibiotics: Peptides of diverse structure and function. *Annual Review of Microbiology*, 61(1):477–501.

Zhang, R., Wang, Y., Yang, Y., Zhang, Y., and Ma, J. (2018). Predicting ctcf-mediated chromatin loops using ctcf-mp. *Bioinformatics (Oxford, England)*, 34(13).

Zwierzyna, M. and Overington, J. P. (2017). Classification and analysis of a large collection of in vivo bioassay descriptions. *PLoS computational biology*, 13(7).

CHAPTER 3. TRANSFER LEARNING IMPROVES ANTIBIOTIC RESISTANCE CLASS PREDICTION

Modified from a manuscript to be submitted soon

Md-Nafiz Hamid[1,2] and Iddo Friedberg[1,2]

[1]Program in Bioinformatics and Computational Biology, Iowa State University, Ames, Iowa, USA

[2]Veterinary Microbiology and Preventive Medicine, Iowa State University, Ames, Iowa, USA.

3.1 Abstract

Antibiotic resistance happens when microbes that could previously be treated with antibiotic drugs are no longer susceptible to them. While antibiotics and antibiotic resistance have always been part of natural microbial physiology, the spread of life threatening antibiotic resistance has been largely due to bacteria gaining resistance conferring genes from other bacteria. Therefore, it is crucial to identify these genes for the purpose of antibiotic resistance surveillance. Predicting the class of antibiotic that a gene confers resistant to can quickly lead to actionable guidance. There are multiple tools available that predict antibiotic resistance class from genes. Yet, there is a lack of benchmarks on the performances of these tools. In this paper, we first developed a dataset that is curated from 15 available databases, and annotated with their antibiotic class labels. We also developed a transfer learning approach with neural networks, TRAC, that outperforms all these tools including a deep learning from scratch model that we developed ourselves. While TRAC provides the current state-of-the-art performance, we hope our newly developed dataset will also provide the community with a much needed standardized dataset to develop novel methods that can predict antibiotic resistance class with superior prediction performance. TRAC is available at https://github.com/nafizh/TRAC.

3.2 Introduction

In the US, each year, at least 2 million people are affected by antibiotic resistance, and at least 23,000 fatally (for Disease Control and Prevention, 2018). In the European Union, around 30,000 people die each year from antibiotic resistance (Cassini et al., 2019). According to a study, without a global response, 10 million people will die from antibiotic resistance infections each year by 2050, and the total GDP loss resulting from antibiotic resistance will be equivalent to 100.2 trillion USD (ONeill et al., 2016). Antibiotic resistance surveillance is an integral aspect of our response towards mitigating clinical and environmental implications of antibiotic resistance. Therefore rapid identification of the antibiotic resistance class from clinical and environmental metagenomic samples is critical. Currently, culture based techniques such as Antimicrobial Susceptibility Testing (AST) are used to detect antibiotic resistance which provides important information to devise patient treatment. While these methods are an important part of the clinical setting, they require microbiology facilities and experts not available at every setting. They also do not work for uncultured bacteria which usually constitute a large part of any complex microbial community (D'Costa et al., 2006). Computational tools can complement these culture based techniques by identifying the antibiotic resistance class from genomic sequences the come from a sample.

Bacteria have always produced antibiotics as secondary metabolites for the purpose of survival (Sengupta et al., 2013). These metabolites are used against other bacteria to gain competitive advantage within the microbial community. In response, other bacteria have developed resistance against these antibiotics through various antibiotic resistance mechanisms (Boolchandani et al., 2019). This phenomena is known as inherent antibiotic resistance. Yet, the rapid spread of antibiotic resistance because of overuse of antibiotics is largely through secondary antibiotic resistance. Secondary antibiotic resistance happens when a bacteria previously susceptible to an antibiotic gains resistance against it by acquiring a resistance conferring gene against an antibiotic from another bacteria. Computational tools can greatly help identify these genes along with the antibiotic class they confer resistant to. Yet, there is a dearth of datasets through which we can judge the relative performance of these computational tools.

In this study, first, we developed a dataset called COALA (COllection of ALl Antibiotic resistance gene databases) from 15 antibiotic resistance databases available online. We also collected the resistance class label for these sequences. We release 3 versions of this dataset. (1) COALA100, the dataset as is, (2) COALA70, the dataset after doing a CD-HIT (Li and Godzik, 2006b) with 0.7 threshold on COALA100 dataset and (3) COALA40, the dataset after doing a CD-HIT with 0.4 threshold on COALA100 dataset.

We then developed a deep neural network based transfer learning approach TRAC (TRansfer learning for Antibiotic resistance gene Classification) that can predict antibiotic classes which resistance genes confer. This can be useful in identifying resistance conferring genes from clinical samples for the purpose of providing a focused drug treatment. Current state-of-the-art tools such as CARD-RGI (Jia et al., 2016), NCBI-AMRFinder(Feldgarden et al., 2019), SARGFAM (Yin et al., 2018) use traditional alignment based methods such as BLAST (Altschul et al., 1990) or HMMER (Potter et al., 2018) or an augmented version of these methods. Recently, a deep learning based approach, DeepARG (Arango-Argoty et al., 2018) was developed that used normalized bit scores as features that were acquired after aligning against known antibiotic resistant genes. In contrast to DeepARG's approach, TRAC tries to predict antibiotic resistance class just from the amino acids of a protein sequence.

Transfer learning is a machine learning technique where a learning algorithm is first trained to solve a task, and then use the trained algorithm to solve another task. Typically, the final task to solve is a separated but related task. The idea is to use the knowledge gained from the first task to boost performance of solving the final task. In contrast, with a typical supervised learning approach, the algorithm is trained to solve a task, and then expected to solve the same task for unseen data in future. This requires ample labeled data — where we know the ground truth for each instance of the training data — to train the supervised learning algorithm. However, in Biology it is quite normal that researchers do not have enough labeled data. Transfer learning allows us to use labeled or more crucially, unlabeled data to train an algorithm, and use that algorithm to train again on the small labeled data. For example, in any kind of image classification task, it is

now common to use a pre-trained model trained on a huge labeled dataset such as Imagenet (Deng et al., 2009), and then again train that model on a smaller labeled dataset of the final task (known as fine-tuning) (Nguyen et al., 2018; Lei et al., 2018). This is an example where a large corpus of labeled data is being used to augment the performance in a task where labeled data is of small amount.

Another example comes from Natural Language Processing (NLP), where it is common these days to use a pre-trained model that has been trained first as a language model on a huge corpus of sentences. These pre-trained models are then used for further downstream supervised classification tasks such as text classification (Howard and Ruder, 2018). Language modeling is the task of assigning a probability to a sentence happening in that language (Ruder et al., 2019). These types of language models are typically trained as to predict the next word in a sentence given some words previously. After a language model learns a probability distribution over all possible sentences in a language, the model is trained on a smaller task specific labeled dataset such as sentiment classification e.g. predict from a given review if the review is positive or negative (Howard and Ruder, 2018).

The Transfer learning approach has been shown to be effective in various tasks of protein sequence classification (Alley et al., 2019; Rives et al., 2019; Bileschi et al., 2019; Rao et al., 2019) where labeled data is of small amount. We have a large amount of unlabeled protein sequences publicly available despite the small amount of labeled data for many crucial tasks, for exmaple, Uniprot TrEMBL (Apweiler et al., 2004) has around 158M protein sequences. In the same vein of the NLP example above, we then leverage this large number of unlabeled protein sequences to learn a language model where we train a deep neural network to predict the next amino acid in a protein sequence. This can be thought of a kind of unsupervised or self-supervised learning where we create a label ourselves from the unlabeled data. For example, in our case, during language model training, the neural network tries to predict the next amino acid in a protein sequence. Here, we have created an artificial supervised task where the label is the next amino acid event though the dataset is unlabeled in its true meaning. By training this way, the neural network learns the

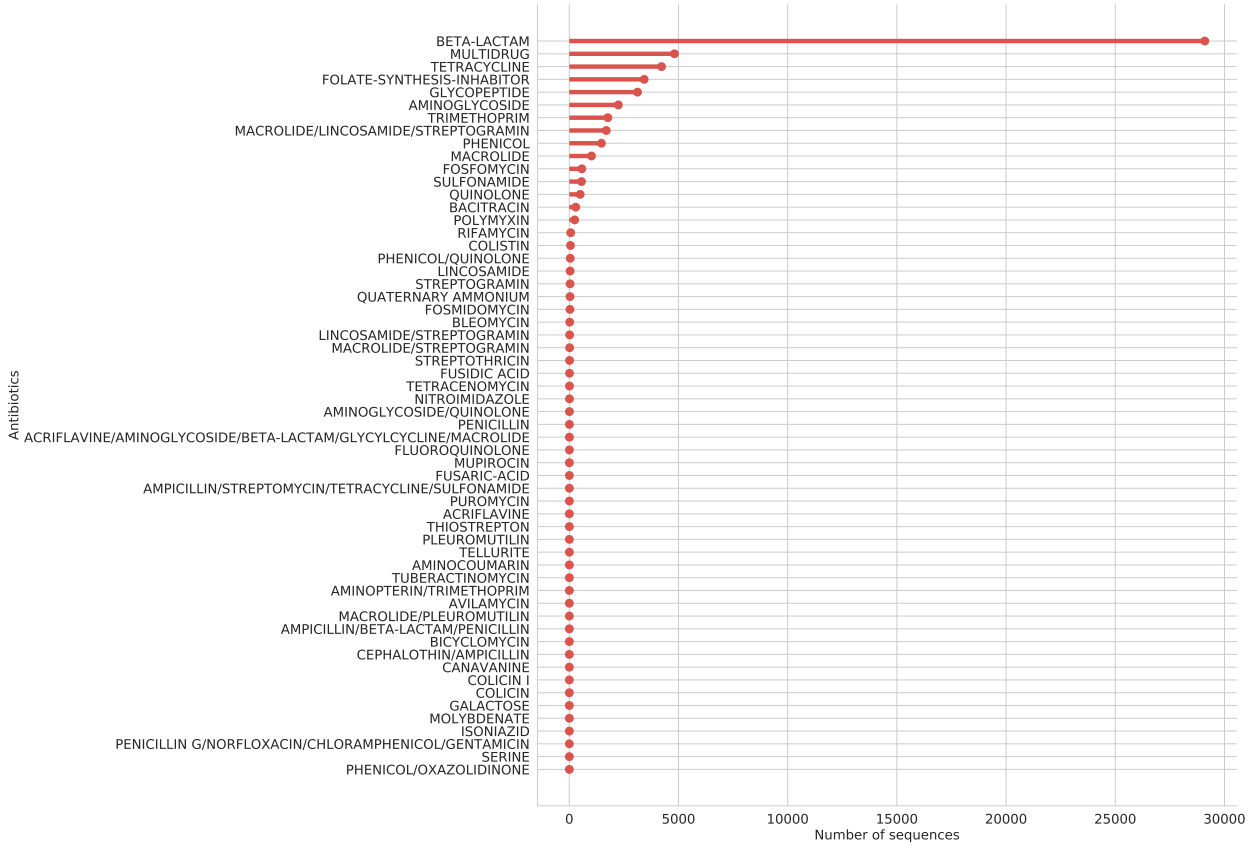
probability distribution of all possible protein sequences. We used the UniRef50 database (Suzek et al., 2014) to train the neural network language model, and then used the trained neural network to predict the antibiotic class the gene confers resistant to by training on a small labeled dataset where the training data are protein sequences, and true labels are antibiotics against which the gene confers resistance. As we will show, TRAC outperforms all state-of-the-art methods by a large margin, and performs better than a deep learning method that was trained only on the small antibiotic resistance labeled dataset.

3.3 Dataset

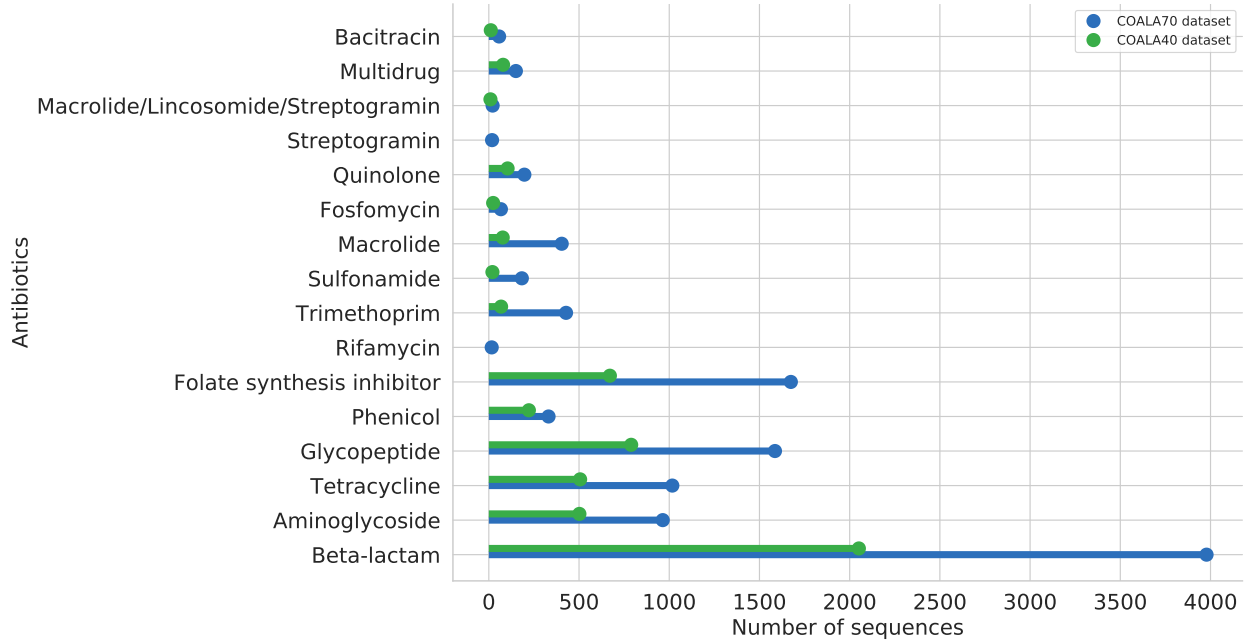
For our unlabeled dataset in the transfer learning approach, we used the UniRef50 database (Suzek et al., 2014) which is a manually curated database of ˜24M protein sequences so that any two sequences do not have more than 50% sequence identity. We only took the bacteria sequences from Uniref50 as our goal is to train on a labeled dataset of antibiotic resistance bacteria protein sequences. This resulted in having 7,34,948 protein sequences. This is our unlabeled dataset to train a neural network on it as a language model.

For the labeled dataset on which we trained and tested TRAC and all other comparison methods, we developed the COALA (COllection of ALl Antibiotic resistance gene databases) database. To get a reliable estimate of the generalization ability of any machine learning task, it is essential that the test set does not provide instances that are identical or similar to instances in the training set. That is why we created two datasets, COALA70 and COALA40, by performing CD-HIT with 0.7 and 0.4 identity threshold on the original COALA dataset. This discards sequences that have higher than 70% and 40% identity respectively for COALA70 and COALA40. We report all models' performances on these two datasets, essentially turning this into a remote homolog detection problem within the context of antibiotic resistance.

We went through all 15 available antibiotic resistance gene databases mentioned in (Boolchandani et al., 2019). These databases are CARD (Jia et al., 2016), ResFinder (Zankari et al., 2012), ResfinderFG (Munk et al., 2018), ARDB (Liu and Pop, 2009), MEGARes (Lakin et al., 2016),



(a) Number and types of antibiotic resistance gene sequences in COALA100 dataset



(b) Number and types of antibiotic resistance gene sequences in COALA70 and COALA40 dataset

Figure 3.1: Different types of antibiotic resistance gene sequences in COALA100, COALA70, and COALA40 dataset. COALA100 dataset has 55491 sequences from 58 classes, COALA70 has 11091 sequences from 16 classes, and COALA40 has 5133 sequences from 14 classes.

NDARO, ARG-ANNOT (Gupta et al., 2014), Mustard (Ruppé et al., 2019), FARME (Wallace et al., 2017), SARG(v2) (Yin et al., 2018), Lahey list of β-lactamases (Bush and Jacoby, 2010), BLDB (Naas et al., 2017), LacED (Thai and Pleiss, 2010; Thai et al., 2009), CBMAR (Srivastava et al., 2014a) , MUBII-TB-DB (Flandrois et al., 2014), and u-CARE (Saha et al., 2015). We collected protein sequences from these databases with their annotations wherever they were available. In total, we collected 55491 protein sequences from 58 antibiotic classes which we named as COALA100. COALA100 has a large number of overlapping sequences. Therefore, we used CD-HIT to reduce redundancy. We used two thresholds of 70% and 40% to develop two databases named COALA70 and COALA40. COALA70 has 11,091 protein sequences from 16 antibiotic classes, and COALA40 has 5133 protein sequences from 14 antibiotic classes. The reduction in protein sequences underscores the redundancy the COALA100 database has.

COALA70 and COALA40 datasets are our small labeled dataset where we have corresponding antibiotic labels for each sequence. We show all models' performances on both the COALA70 and COALA40 dataset.

3.4 TRAC

To use transfer learning, we applied the ULMFit (Universal Language Model Fine-tuning) (Howard and Ruder, 2018) approach used in natural language processing to perform various text classification tasks. ULMFit uses the AWD-LSTM (Averaged stochastic gradient descent Weight-Dropped LSTM) (Merity et al., 2017) architecture to train a language model on a large English language corpus. This step is the General-domain language model pre-training. This trained language model is then trained on the small labeled dataset on which the final text classification task will be performed. This step is know as language model fine-tuning. Finally, the architecture is modified to do classification on the labeled dataset.

In our case, we used the AWD-LSTM architecture to train a neural network on the UniRef50 Bacteria sequences as a language model (Figure 3.2a). The architecture takes each amino acid represented as an embedding vector as input. It has 3 layers of LSTM (Long Short Term Memory)

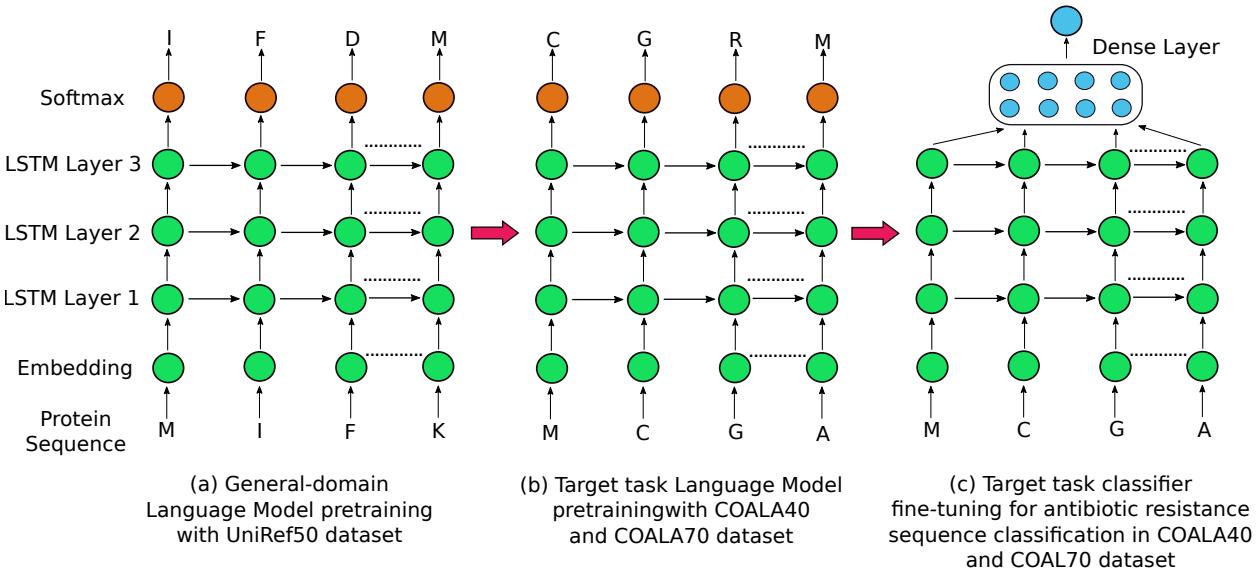


Figure 3.2: Different stages of training for TRAC. (a) We train the neural network on the UniRef50 dataset where the model is trying to predict the next amino acid in a protein sequence. (b) This is the finetuning step of the pretraining where we train the model on COALA70 and COALA40 protein sequences to predict the next amino acid. (c) The final classification task where the pretrained model is now trained in a supervised way to do a multiclass classification of antibiotic resistance classes.

(Hochreiter and Schmidhuber, 1997) units. LSTMs are special recurrent neural networks that can capture long range dependency in sequential data. The AWD-LSTM model tries to predict the next amino acid in the protein sequence. The output layer is a softmax layer where the model assigns a probability to each amino acid of being the next amino acid in the protein sequence. This parallels the General-domain language model pretraining. Next, we train a target task language model (Figure 3.2b) pretraining over the COALA40 dataset before doing classification training on COALA40, and this step parallels the language model fine tuning step in ULMFit. The target task language model is trained similarly like the general domain language model in that we train the model to predict the next amino acid in a protein sequence but on the small labeled dataset which will be used for the final classification task. The language model fine-tuning is also done for the COALA70 dataset.

Finally, We took this pretrained language model, and used it to classify protein sequences into antibiotics they are resistant to. For both COALA40 and COALA70 datasets, we report the accuracy over a 10-fold cross validation. For both pretraining and classification, a 1-cycle learning rate schedule (Smith, 2018) with the AdamW (Loshchilov and Hutter, 2017) optimizer was used. For pretraining, categorical cross entropy, and for classification label smoothing cross entropy (Szegedy et al., 2016) was used as loss function.

3.5 Methods for comparison

We use two criteria to select methods to which we compare TRAC. First, the method has to predict antibiotic resistance classes from protein sequences (not DNA/RNA reads). Second, the method has to be available as a software or as a web tool. We looked at all the methods mentioned in (Boolchandani et al., 2019). Four current state-of-the-art methods - CARD-RGI, NCBI-AMRFinder, DeepARG and SARGFAM match these two criteria. Besides these four methods, we developed a deep learning method that is only trained on the smaller labeled dataset as well as a Random Forest (Breiman, 2001) machine learning method. The deep learning method was developed to verify whether training on an unlabeled dataset before the final classification task truly improves the performance on a labeled dataset. The Random Forest model was developed to verify whether deep learning and transfer learning methods perform better than traditional machine learning approaches. All six of these methods were tested on the COALA70 and COALA40 datasets.

CARD-RGI CARD-RGI (Jia et al., 2016) integrates an antibiotic resistance ontology with homology and Single Neucleotide Polymorphism (SNP) models, and uses these models with a curated set of antimicrobial resistance sequences to identify antibiotic resistance from protein or neucleotide data.

NCBI-AMRFinder NCBI-AMRFinder(Feldgarden et al., 2019) developed from NCBI uses BLASTP (Altschul et al., 1990) and HMMER (Potter et al., 2018) against the NDARO (Feldgarden et al., 2019) database to detect antibiotic resistance from in novel protein sequences.

SARGFAM SARGFAM(Yin et al., 2018) is a collection of profile HMMs built from more than 12,000 antibiotic resistance genes collected from CARD, ARDB and NCBI-NR databases. It runs a hmmsearch against these profile HMMs to detect antibiotic resistance in novel protein sequences.

DeepARG DeepARG(Arango-Argoty et al., 2018) is a deep learning (LeCun et al., 2015) based method to classify protein sequences into the antibiotics they are resistant against. The authors created a dataset of antibiotic resistance genes from CARD, ARDB and UNIPROT. Sequences from UNIPROT were aligned against CARD and ARDB with Diamond (Buchfink et al., 2015), and the normalized bit scores from the alignment were used as features for the deep learning model.

Deep Learning (Ours) We developed a deep learning model that was trained on the COALA70 and COALA40 datasets. We used a self-attention based sentence embedding model introduced in (Lin et al., 2017). Figure 4.1 shows a schematic of the architecture of the model we used for antibiotic resistant gene classification. For input, we represented each amino acid in a protein sequence as an embedding vector which was randomly initialized, and then trained end-to-end. We used an LSTM (Hochreiter and Schmidhuber, 1997) recurrent neural network which takes these embedding vectors as input. Following that is the self-attention part which we can think of as a feed-forward neural network with one hidden layer. This network takes the output from the LSTM layer as input, and outputs weights which are also called attentions. We multiplied the outputs of the LSTM layer with these attentions to get a weighted view of the LSTM hidden states. This multiplication result gets passed onto the next layer of the network, and the final layer is a softmax layer which assigns a probability to each class of antibiotics so that they sum to 1. In training, AdamW (Loshchilov

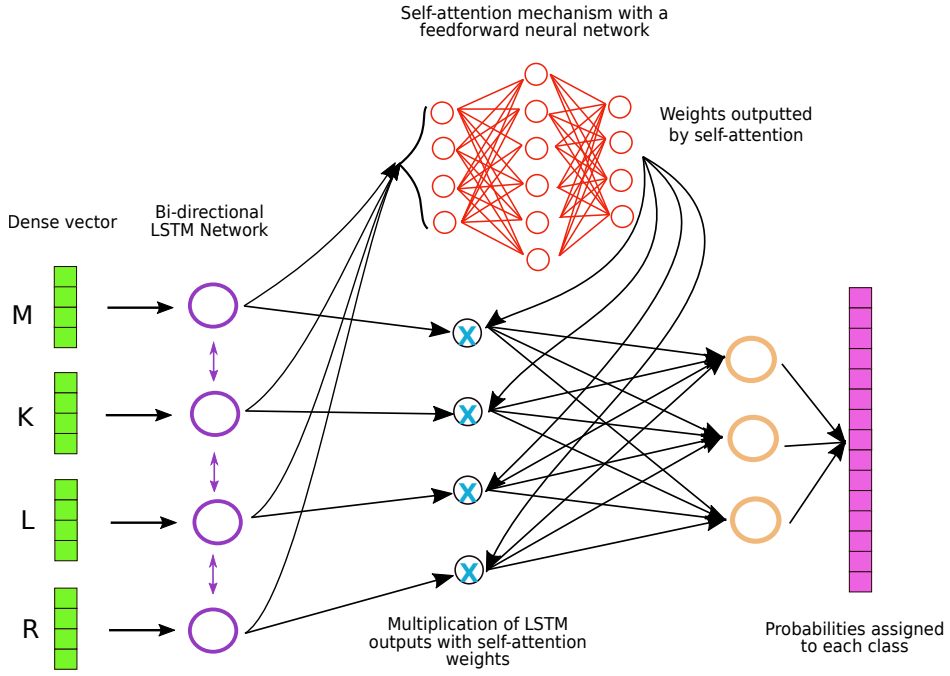


Figure 3.3: Our self-attention model used for antibiotic resistance gene classification. Each amino acid in a protein sequence is represented by a vector. These vectors are used as input for LSTM neural networks (Hochreiter and Schmidhuber, 1997). Outputs of the LSTM network goes into a simple feed-forward network, and the output of this feed-forward network are weights which sum to 1. These weights are multiplied with the outputs of the LSTM network to give them weights. This mechanism lets the model learn which input from the protein sequence is crucial in classifying a gene into its correct antibiotic class. Finally, there is a softmax layer, the size of which is corresponds to the number of antibiotic classes we have. This layer assigns a probability to each class. The class that is assigned the highest probability becomes the prediction for a certain protein sequence.

and Hutter, 2017) optimizer with a 1-cycle learning rate schedule (Smith, 2018) with a negative loss likelihood loss function was used.

Random Forest (Ours) We applied a random forest model on the COALA70 and COALA40 datasets, and measure the performance. For features, we used the simple and popular k-mer count for each protein sequences. The hyper-parameters of the random forest model were tuned with random search using the popular Scikit-learn(Pedregosa et al., 2011) machine learning library.

3.6 Results

Figure 3.4 shows a comparison between all the models on both COALA40 and COALA70 datasets. We use the accuracy metric to compare the models. For a set of predictions $\hat{Y} = (\hat{y_1}, \hat{y_3}, ...\hat{y_N})$ and a set of true labels $Y = (y_1, y_3, ...y_N)$ accuracy is defined as:

$$Accuracy(Y, \hat{Y}) = \frac{1}{N} \sum_{n=1}^{N} I(y = \hat{y}) \tag{3.1}$$

where I is an indicator function which is defined as:

$$I(y, \hat{y}) = \begin{cases} 1, & if \quad y = \hat{y} \\ 0, & otherwise \end{cases}$$

The accuracy is the mean of 10-fold cross validation on both COALA40 and COALA70 datasets. The hyper-parameters were chosen by nested cross validation so that we do not overestimate on the final test set. For COALA40 dataset, we can see from the figure that TRAC outperforms all other models with an accuracy of 0.52, and the next closest performing model is our own deep learning model that we constructed without any pretraining (accuracy 0.43). Random forest performs almost with a similar accuracy of 0.41. CARD-RGI (accuracy 0.26), DeepARG (accuracy 0.12), SARGFAM (accuracy 0.08) and NCBI-AMRFinder (accuracy 0.06) are behind in terms of performance from the top 3 methods. In the COALA70 dataset, TRAC again outperforms all other methods by a large margin (accuracy 0.69). Interestingly, again a well tuned random forest model (accuracy 0.61) performs similarly to a deep learning model (accuracy 0.60) that was not pretrained. With a bigger dataset of COALA70, performance of CARD-RGI comparatively increases (accuracy 0.54). In contrast, DeepARG, SARGFAM and NCBI AMRFinder still lags behind by some distance. Performance of all the models on both COALA40 and COALA70 dataset can also be seen in Table 3.1.

3.7 Discussion

Identifying genes that confer resistance to antibiotics is crucial to track their presence in metagenomic and clinical samples. Current techniques such as AST takes a long time to come up with

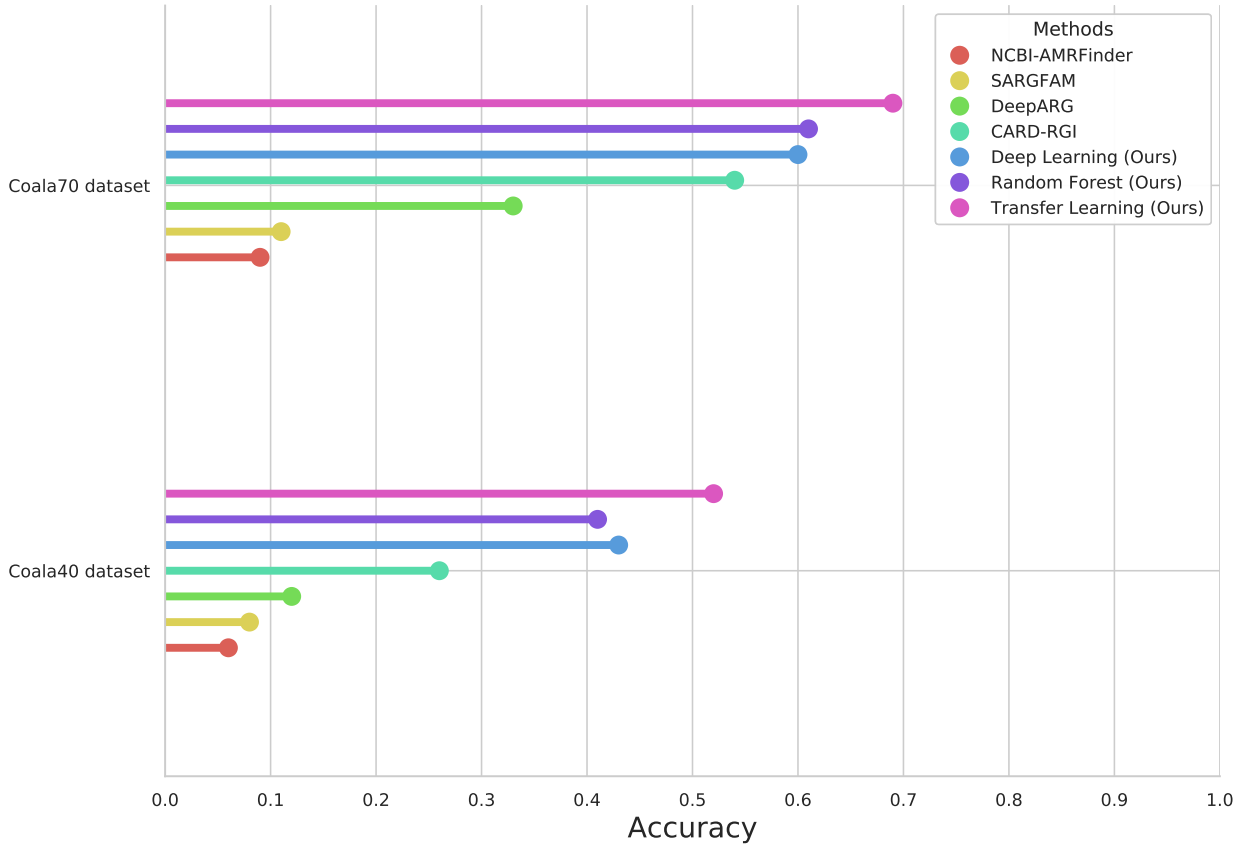


Figure 3.4: Comparison of performances of all models in terms of mean accuracy from 10-fold nested cross validation on both COALA70 and COALA40 datasets.

results as well as requires microbiology facilities and expertise which might not be scalable. Computational tools such as bioinformatics and machine learning can aid human experts in these settings. In this study, we compare currently available computational tools to predict antibiotic class that a gene might confer resistance to. Towards this end, we developed a dataset, COALA, that contains protein sequences from 15 antibiotic resistance databases available online. We also curated the antibiotic resistance class labels for these protein sequences from their respective databases. This resulted in a dataset that has huge number of redundant and similar sequences. To gain reliable estimate of how well all the computational methods perform, we trim down this dataset by CD-HIT with 40% and 70% identity threshold, resulting in two datasets, COALA40 and COALA70. This essentially turns this problem into a remote homolog prediction problem where we are trying to gauge how good the models are in terms of finding sequence specificity for resistance conferring functionality.

We can see from the performances of all models that TRAC, the transfer learning approach performs better by a great distance in terms of accuracy. Even so, TRAC gains around 69% accuracy for COALA70 and around 52% accuracy for COALA40. This indicates that multi-class classification for antibiotic resistance class prediction is indeed a difficult task. However, the alignment free approaches — TRAC, Deep learning without pretraining, and random forest all perform better than the other 4 methods, namely, NCBI-AMRFinder, SARGFAM, CARD-RGI, DeepARG which depend on alignment based approaches such as BLAST or HMMER completely or partially. This is crucial with respect to the fact that performing CD-HIT with such low thresholds of identity turned this problem into a remote homolog prediction problem. Within the alignment free approaches, surprisingly, a deep learning model without pretraining did not perform better than a well tuned random forest model that was trained on k-mer features. This leads to the question, why TRAC is performing significantly better than the deep learning without pretraining model. We hypothesize that by training on unlabeled bacteria sequences and trying to predict the next amino acid in a sequence, TRAC gains an internal representation of bacterial sequences which helps it to identify remote homologs in the downstream classification task. Another possible reason is the labeled

datasets, especially, COALA40 do not have enough training sequences for a deep learning model to learn without pretraining. TRAC mitigates this problem by shifting the burden of learning onto the unlabeled dataset for pretraining.

We hope therefore COALA70 and COALA40 datasets will provide good test beds for developing novel machine learning methods with superior accuracy performance equivalent to that provided by the MNIST dataset (LeCun et al., 1998) for the deep learning community. The performance of TRAC provides further proof that language model pretraining boosts model performance with few labeled data (Strodthoff et al., 2019) which is the case in many bioinformatics tasks. For future work, it will be interesting to unpack the internal representation of the protein sequences learnt by TRAC, and how it compares with traditional hand crafted bioinformatics features such as Position-Specific-Scoring-Matrices (PSSM). We are also excited with the prospects of integrating latest advancements in the Natural language field such as Transformer (Vaswani et al., 2017) inspired encoding techniques (e.g. BERT (Devlin et al., 2018)) into the pretraining step, and improving antibiotic resistance class prediction towards complementing current clinical practices.

3.8 Appendix: supplementary material

Table 3.1: Mean accuracy from 10-fold nested cross validation for all methods over both COALA40 and COALA70 datasets. Standard deviation of accuracy is shown for our models where cross validation is done.

	COALA40 dataset	**COALA70 dataset**
CARD-RGI	0.261	0.544
NCBI-AMRFinder	0.062	0.093
SARGFAM	0.085	0.110
DeepARG	0.126	0.330
Random Forest (Ours)	0.416 ± 0.007	0.614 ± 0.065
Deep Learning (Ours)	0.436 ± 0.018	0.609 ± 0.097
Transfer Learning (Ours)	**0.520 ± 0.084**	**0.696 ± 0.078**

3.9 References

Alley, E. C., Khimulya, G., Biswas, S., AlQuraishi, M., and Church, G. M. (2019). Unified rational protein engineering with sequence-only deep representation learning. *bioRxiv*, page 589333.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

Apweiler, R., Bairoch, A., Wu, C. H., Barker, W. C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., et al. (2004). Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl 1):D115–D119.

Arango-Argoty, G., Garner, E., Pruden, A., Heath, L. S., Vikesland, P., and Zhang, L. (2018). Deeparg: a deep learning approach for predicting antibiotic resistance genes from metagenomic data. *Microbiome*, 6(1):23.

Bileschi, M. L., Belanger, D., Bryant, D. H., Sanderson, T., Carter, B., Sculley, D., DePristo, M. L., and Colwell, L. J. (2019). Using deep learning to annotate the protein universe. *bioRxiv*, page 626507.

Boolchandani, M., DSouza, A. W., and Dantas, G. (2019). Sequencing-based methods and resources to study antimicrobial resistance. *Nature Reviews Genetics*, page 1.

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Buchfink, B., Xie, C., and Huson, D. H. (2015). Fast and sensitive protein alignment using diamond. *Nature methods*, 12(1):59.

Bush, K. and Jacoby, G. A. (2010). Updated functional classification of β-lactamases. *Antimicrobial agents and chemotherapy*, 54(3):969–976.

Cassini, A., Högberg, L. D., Plachouras, D., Quattrocchi, A., Hoxha, A., Simonsen, G. S., Colomb-Cotinat, M., Kretzschmar, M. E., Devleesschauwer, B., Cecchini, M., et al. (2019). 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.

D'Costa, V. M., McGrann, K. M., Hughes, D. W., and Wright, G. D. (2006). Sampling the antibiotic resistome. *Science*, 311(5759):374–377.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Feldgarden, M., Brover, V., Haft, D. H., Prasad, A. B., Slotta, D. J., Tolstoy, I., Tyson, G. H., Zhao, S., Hsu, C.-H., McDermott, P. F., et al. (2019). Using the ncbi amrfinder tool to determine antimicrobial resistance genotype-phenotype correlations within a collection of narms isolates. *BioRxiv*, page 550707.

Flandrois, J.-P., Lina, G., and Dumitrescu, O. (2014). Mubii-tb-db: a database of mutations associated with antibiotic resistance in mycobacterium tuberculosis. *BMC bioinformatics*, 15(1):107.

for Disease Control, C. and Prevention (2018). About Antimicrobial Resistance. https://www.cdc.gov/drugresistance/about.html.

Gupta, S. K., Padmanabhan, B. R., Diene, S. M., Lopez-Rojas, R., Kempf, M., Landraud, L., and Rolain, J.-M. (2014). Arg-annot, a new bioinformatic tool to discover antibiotic resistance genes in bacterial genomes. *Antimicrobial agents and chemotherapy*, 58(1):212–220.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Howard, J. and Ruder, S. (2018). Universal language model fine-tuning for text classification. *arXiv preprint arXiv:1801.06146*.

Jia, B., Raphenya, A. R., Alcock, B., Waglechner, N., Guo, P., Tsang, K. K., Lago, B. A., Dave, B. M., Pereira, S., Sharma, A. N., et al. (2016). Card 2017: expansion and model-centric curation of the comprehensive antibiotic resistance database. *Nucleic acids research*, page gkw1004.

Lakin, S. M., Dean, C., Noyes, N. R., Dettenwanger, A., Ross, A. S., Doster, E., Rovira, P., Abdo, Z., Jones, K. L., Ruiz, J., et al. (2016). Megares: an antimicrobial resistance database for high throughput sequencing. *Nucleic acids research*, 45(D1):D574–D580.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

LeCun, Y., Cortes, C., and Burges, C. J. (1998). The mnist database of handwritten digits, 1998. *URL http://yann. lecun. com/exdb/mnist*, 10:34.

Lei, H., Han, T., Zhou, F., Yu, Z., Qin, J., Elazab, A., and Lei, B. (2018). A deeply supervised residual network for hep-2 cell classification via cross-modal transfer learning. *Pattern Recognition*, 79:290–302.

Li, W. and Godzik, A. (2006). Cd-hit: a fast program for clustering and comparing large sets of protein or nucleotide sequences. *Bioinformatics*, 22(13):1658–1659.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv:1703.03130 [cs]*. arXiv: 1703.03130.

Liu, B. and Pop, M. (2009). Ardbantibiotic resistance genes database. nucleic acids res37 (suppl 1): D443–d447.

Loshchilov, I. and Hutter, F. (2017). Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*.

Merity, S., Keskar, N. S., and Socher, R. (2017). Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*.

Munk, P., Knudsen, B. E., Lukjancenko, O., Duarte, A. S. R., Van Gompel, L., Luiken, R. E., Smit, L., Schmitt, H., Garcia, A. D., Hansen, R. B., et al. (2018). Abundance and diversity of the faecal resistome in slaughter pigs and broilers in nine european countries. *Nat Microbiol*, 3(8):898–908.

Naas, T., Oueslati, S., Bonnin, R. A., Dabos, M. L., Zavala, A., Dortet, L., Retailleau, P., and Iorga, B. I. (2017). Beta-lactamase database (bldb)–structure and function. *Journal of enzyme inhibition and medicinal chemistry*, 32(1):917–919.

Nguyen, L. D., Lin, D., Lin, Z., and Cao, J. (2018). Deep cnns for microscopic image classification by exploiting transfer learning and feature concatenation. In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5. IEEE.

ONeill, J., Davies, S., Rex, J., White, L., Murray, R., et al. (2016). Review on antimicrobial resistance, tackling drug-resistant infections globally: final report and recommendations. *London: Wellcome Trust and UK Government*.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Potter, S. C., Luciani, A., Eddy, S. R., Park, Y., Lopez, R., and Finn, R. D. (2018). Hmmer web server: 2018 update. *Nucleic acids research*, 46(W1):W200–W204.

Rao, R., Bhattacharya, N., Thomas, N., Duan, Y., Chen, X., Canny, J., Abbeel, P., and Song, Y. S. (2019). Evaluating protein transfer learning with tape. *arXiv preprint arXiv:1906.08230*.

Rives, A., Goyal, S., Meier, J., Guo, D., Ott, M., Zitnick, C. L., Ma, J., and Fergus, R. (2019). Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *bioRxiv*, page 622803.

Ruder, S., Peters, M. E., Swayamdipta, S., and Wolf, T. (2019). Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18.

Ruppé, E., Ghozlane, A., Tap, J., Pons, N., Alvarez, A.-S., Maziers, N., Cuesta, T., Hernando-Amado, S., Clares, I., Martínez, J. L., et al. (2019). Prediction of the intestinal resistome by a three-dimensional structure-based method. *Nature microbiology*, 4(1):112.

Saha, S. B., Uttam, V., and Verma, V. (2015). u-care: user-friendly comprehensive antibiotic resistance repository of escherichia coli. *Journal of clinical pathology*, 68(8):648–651.

Sengupta, S., Chattopadhyay, M. K., and Grossart, H.-P. (2013). The multifaceted roles of antibiotics and antibiotic resistance in nature. *Frontiers in microbiology*, 4:47.

Smith, L. N. (2018). A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. *arXiv preprint arXiv:1803.09820*.

Srivastava, A., Singhal, N., Goel, M., Virdi, J. S., and Kumar, M. (2014). Cbmar: a comprehensive β-lactamase molecular annotation resource. *Database*, 2014.

Strodthoff, N., Wagner, P., Wenzel, M., and Samek, W. (2019). Udsmprot: Universal deep sequence models for protein classification. *bioRxiv*, page 704874.

Suzek, B. E., Wang, Y., Huang, H., McGarvey, P. B., Wu, C. H., and Consortium, U. (2014). Uniref clusters: a comprehensive and scalable alternative for improving sequence similarity searches. *Bioinformatics*, 31(6):926–932.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2818–2826.

Thai, Q. K., Bös, F., and Pleiss, J. (2009). The lactamase engineering database: a critical survey of tem sequences in public databases. *BMC genomics*, 10(1):390.

Thai, Q. K. and Pleiss, J. (2010). Shv lactamase engineering database: a reconciliation tool for shv β-lactamases in public databases. *BMC genomics*, 11(1):563.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Wallace, J. C., Port, J. A., Smith, M. N., and Faustman, E. M. (2017). Farme db: a functional antibiotic resistance element database. *Database*, 2017.

Yin, X., Jiang, X.-T., Chai, B., Li, L., Yang, Y., Cole, J. R., Tiedje, J. M., and Zhang, T. (2018). Args-oap v2. 0 with an expanded sarg database and hidden markov models for enhancement characterization and quantification of antibiotic resistance genes in environmental metagenomes. *Bioinformatics*, 34(13):2263–2270.

Zankari, E., Hasman, H., Cosentino, S., Vestergaard, M., Rasmussen, S., Lund, O., Aarestrup, F. M., and Larsen, M. V. (2012). Identification of acquired antimicrobial resistance genes. *Journal of antimicrobial chemotherapy*, 67(11):2640–2644.

CHAPTER 4. ENHANCED RELIABILITY FOR OUT OF DISTRIBUTION DATA DETECTION IN CLASSIFYING ANTIBIOTIC RESISTANCE GENES

Modified from a manuscript to be submitted soon

Md-Nafiz Hamid[1,2] and Iddo Friedberg[1,2]

[1]Program in Bioinformatics and Computational Biology, Iowa State University, Ames, Iowa, USA

[2]Veterinary Microbiology and Preventive Medicine, Iowa State University, Ames, Iowa, USA.

4.1 Abstract

Antibiotic resistance monitoring is of paramount importance in the face of this ongoing global epidemic. Using traditional alignment based methods to detect antibiotic resistant genes result in huge number of false negatives. In this paper, we introduce a deep learning model based on a self-attention architecture that can classify antibiotic resistant genes into correct classes with high precision and recall only taking protein sequences as input. Additionally, deep learning models trained with traditional optimization algorithms (e.g. Adam, SGD) provide poor posterior estimates when tested against Out-of-Distribution (OoD) antibiotic resistant/non-resistant genes. We train our model with an optimization method called Preconditioned Stochastic Gradient Langevin Dynamics (pSGLD) which provides reliable uncertainty estimates when tested against OoD data. All code for the project can be found at https://github.com/nafizh/OoD_AMR.

4.2 Introduction

Antibiotic resistance is a global scourge that is taking an increasing toll in mortality and morbidity, in both nosocomial and community acquired infections(Neu, 1992; Organization, 2014). A growing number of once easily treatable infectious diseases such as tuberculosis, gonorrhea, and pneumonia are becoming harder to treat as the scope of effective drugs is shrinking. The CDC

estimates that 2,000,000 illnesses and 23,000 people die annually from antibiotic resistance in the US alone(for Disease Control and Prevention, 2018). The overuse of antibiotics in health care and agriculture is exacerbating the problem to the point that the World Health Organization is considering antibiotic resistance "one of the biggest threats to global health, food security and human development today". Identifying genes associated with antibiotic resistance is an important first step towards dealing with the problem(Brown and Wright, 2016), and providing a narrow-spectrum treatment, targeted solely against the types of resistance displayed. This statement is especially true when dealing with genes acquired from human or environmental metagenomic samples(Perry et al., 2014). A rapid identification of the class of antibiotic resistance that may exist in a given environmental or clinical microbiome sample can provide immediate guidance to treatment and prevention.

In this study, we developed a deep neural network that can predict antibiotic resistance genes into 15 classes of antibiotics from protein sequences as well as detect Out-of-Distribution (OoD) data — genes that confer resistance against antibiotic classes that were not included during training, or non-antibiotic resistance genes. This property can be useful in identifying metagenomic sample with resistance conferring genes for the purpose of providing a focused drug treatment. Traditional methods (Kleinheinz et al., 2014; Davis et al., 2016; Pal et al., 2016) to identify antibiotic resistance genes usually take an alignment based best-hit approach which causes the methods to produce many false negatives (Arango-Argoty et al., 2018). Recently, a deep learning based approach was developed that uses normalized bit scores as features that were acquired after aligning against known antibiotic resistance genes (Arango-Argoty et al., 2018). In contrast, our model only uses the amino acids of protein sequence as its input, and as such doesn't need do any kind of alignment against a set database.

However, neural networks are known for providing high confidence scores on inputs that are from a different probability distribution than what the model was trained on (Palacci and Hess, 2018; Choi and Jang, 2018). This can result in disastrous consequences in sensitive applications such as health care or self-driving systems. Here, we develop deep learning models trained with

an optimization method not commonly used called Preconditioned Stochastic Gradient Langevin Dynamics (pSGLD) (Li et al., 2016) as well as a traditional optimization method known as ADAM (Kingma and Ba, 2014). Both models provide significant accuracy on the test set in terms of predicting antibiotic resistance solely from the protein sequence. But we show that the model trained with pSGLD is better equipped to predict OoD data i.e., it assigns a low probability to sequences from proteins that are not related to antibiotic resistance or are from antibiotic classes that were not included in training.

4.3 Detection of Out-of-distribution data with Preconditioned Stochastic Gradient Langevin Dynamics (pSGLD)

Typically, neural networks are trained with optimization methods such as Stochastic gradient descent (SGD) (Robbins and Monro, 1985) or its variants such as Adam (Kingma and Ba, 2014), Adagrad (Duchi et al., 2011), RMSprop (Tieleman and Hinton, 2012) etc. In SGD, for each iteration a mini-batch from the dataset is used to update the parameters of the neural network. For each iteration t, training data $X_t = \{x_{t1}, ..., x_{tn}\}$ is provided, and for parameters θ, the update $\Delta\theta_t$ is:

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{ti}|\theta_t)\right) \tag{4.1}$$

However, SGD or its variants do not capture parameter uncertainty. In contrast, Bayesian approaches such as Markov Chain Monte Carlo (MCMC) (Robert and Casella, 2004) techniques do capture uncertainty estimates. One such class of techniques are Langevin dynamics (Roberts and Stramer, 2002) which inject Gaussian noise into Equation 4.1 so that the parameters do not collapse into the Maximum a posteriori (MAP) solution:

$$\Delta\theta_t = \frac{\epsilon}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{ti}|\theta_t)\right) + \eta_t, where, \eta_t \sim N(0, \epsilon) \tag{4.2}$$

MCMC techniques require that the algorithm go over the entire dataset per iteration before making a parameter update. This slows down the model training process, and also requires huge computational costs. To remove this problem, Stochastic Gradient Langevin Dynamics (SGLD)

was introduced (Welling and Teh, 2011), which combined the best of both worlds i.e. inserting Gaussian noise into each mini-batch of training data. In SGLD, during each iteration for SGD, Gaussian noise is injected which has a variance of the step-size ϵ_t:

$$\Delta\theta_t = \frac{\epsilon_t}{2}\left(\nabla \log p(\theta_t) + \frac{N}{n}\sum_{i=1}^{n} \nabla \log p(x_{ti}|\theta_t)\right) + \eta_t, where, \eta_t \sim N(0, \epsilon_t) \tag{4.3}$$

This injection of Gaussian noise has an advantageous side-effect as it also provides a better calibration of confidence scores of predictions on OoD data. For example, (Palacci and Hess, 2018) showed that an SGLD trained neural network provides low confidence scores when trained on the MNIST (LeCun et al., 2010) dataset (a dataset of images of numbers) but tested on the NotMNIST (a dataset of images of letters) dataset (Bulatov, 2011); whereas an SGD trained neural network still naively provides high confidence scores when tested on NotMNIST dataset.

We used preconditioned SGLD (pSGLD) (Li et al., 2016) - a variation of SGLD where we introduce noise to RMSprop, a modification of the SGD algorithm - to train a neural network to classify protein sequences into their resistant antibiotic classes. In the experiment section, we show that a pSGLD trained network provides really high accuracy when classifying antibiotic resistance genes for the classes it was trained on while providing low confidence scores when doing prediction on OoD protein sequences. In contrast, an ADAM trained model provides high accuracy when classifying antibiotic resistance genes for the classes it was trained on as well as when doing prediction on OoD protein sequences.

4.4 Experimental setup

Dataset We used a modified version of the dataset curated in the DeepARG study (Arango-Argoty et al., 2018). Briefly, The dataset was created from the CARD (Jia et al., 2016), ARDB (Liu and Pop, 2008) and UNIPROT (Consortium et al., 2018) databases with a combination of computational and manual curation. The original dataset has 14974 protein sequences that are resistant to 34 different antibiotics (our classes in the multi-class classification task). There were

19 classes that had training samples of 11 sequences or less. We discarded these classes and were left with 15 classes with a total of 14907 protein sequences.

Model We used a self-attention based sentence embedding model introduced in (Lin et al., 2017). Figure 4.1 shows a schematic of the architecture of the model we used for antibiotic resistant gene classification. For input, we represented each amino acid in a protein sequence as an embedding vector which was randomly initialized, and then trained end-to-end. We used Long short term memory (LSTM) (Hochreiter and Schmidhuber, 1997) recurrent neural network which takes these embedding vectors as input. Following that is the self-attention part which we can think of as a feed-forward neural network with one hidden layer. This network takes the output from the LSTM layer as input, and outputs weights which are also called attentions. We multiplied the outputs of the LSTM layer with these attentions to get a weighted view of the LSTM hidden states. This multiplication result gets passed onto the next layer of the network, and the final layer is a softmax layer which assigns a probability to each of the 15 classes of antibiotics that add upto 1.

The model that is trained with Adam has one bi-directional LSTM layer of size 64. The embeddings for each amino acid are of size 10. Dropout of 0.4 was used for regularization. We used a learning rate of 0.001 with a weight decay value of 0.0001 for ADAM. For pSGLD, the model has one bi-directional LSTM layer of size 128. Embedding vectors for amino acids are of size 400. A learning rate of 0.001, and dropout of 0.4 was also used for pSGLD.

We divided our dataset into a 70/20/10% training, validation, and test set split. We trained our model with Adam and pSGLD on the training dataset, and tuned the hyper-parameters by checking the performance on the validation dataset. We report both models' performances on the test set.

4.5 Results

Figure 4.2 shows the performance of Adam and pSGLD trained models on the test set in terms of Precision, Recall and F_1 for each class and overall.

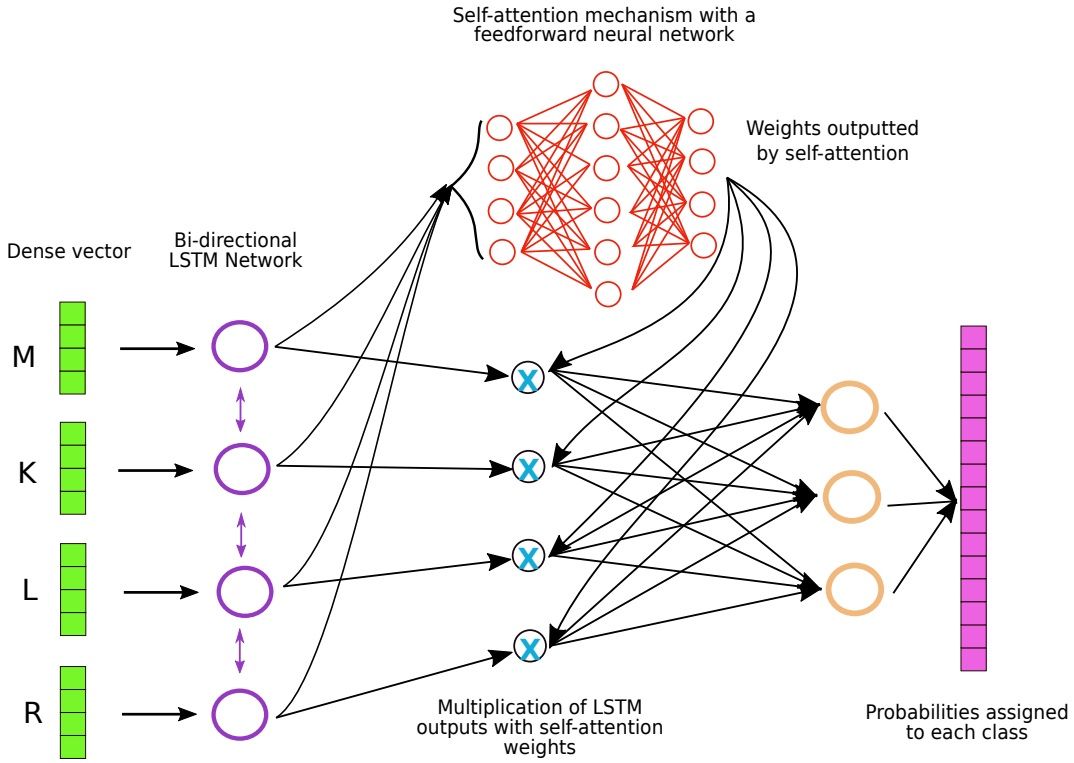
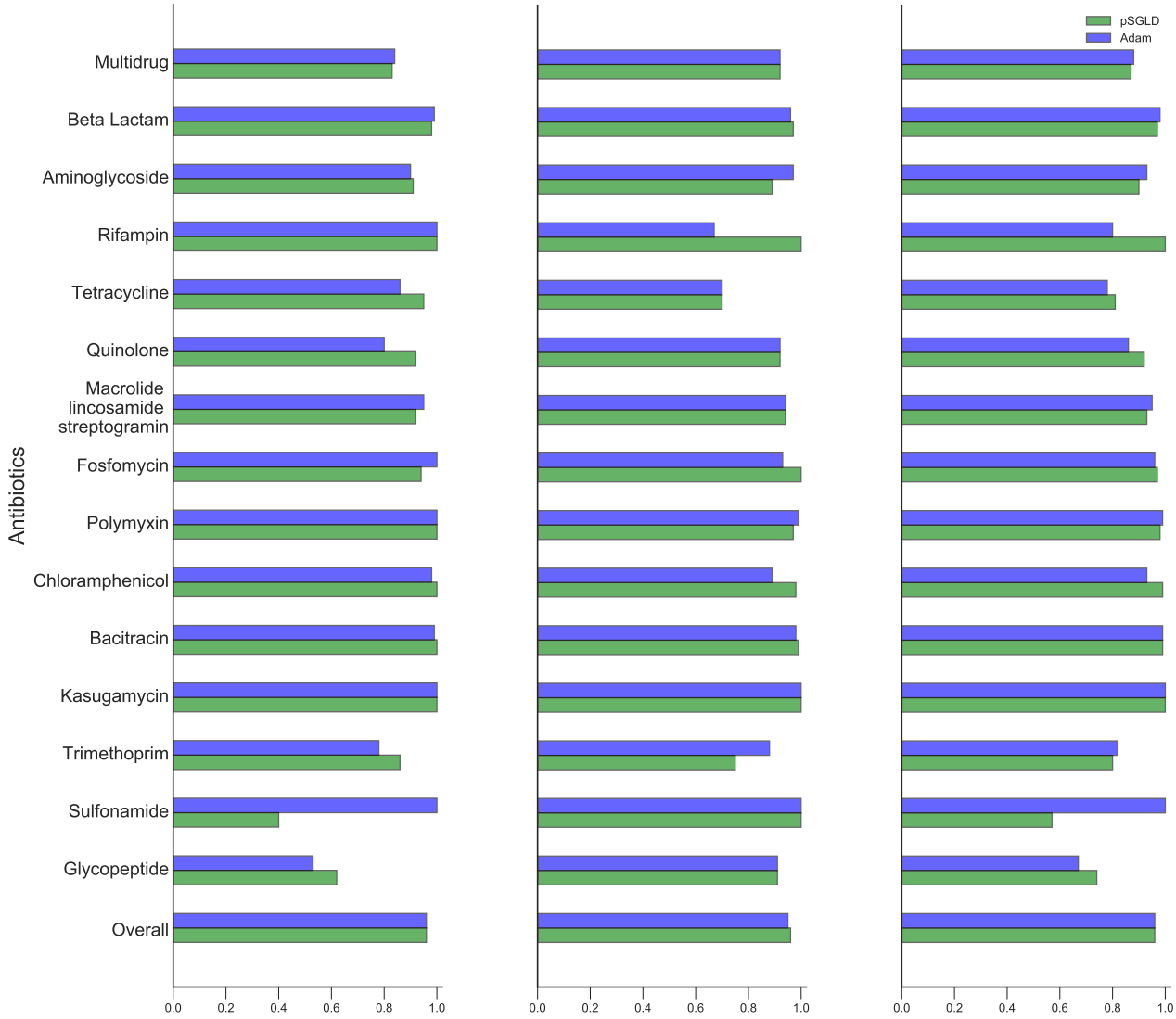
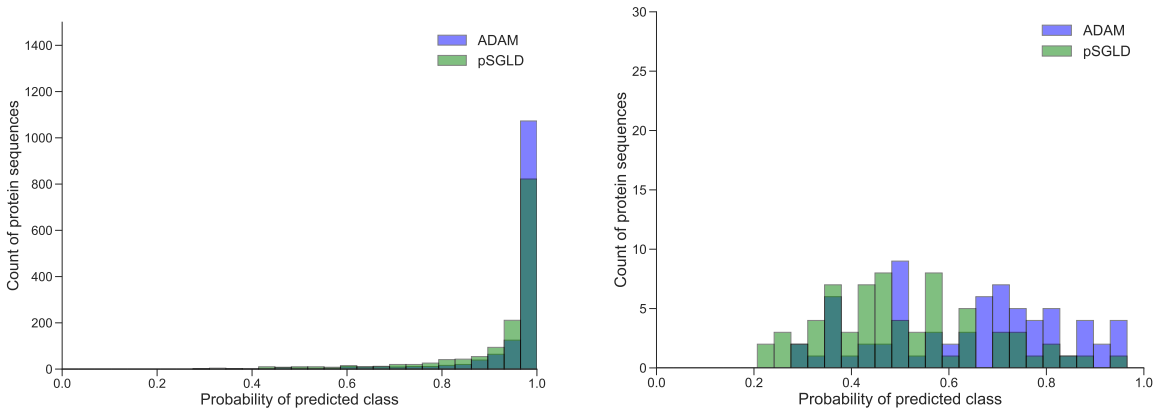


Figure 4.1: Our self-attention model used for antibiotic resistance gene classification. Each amino acid in a protein sequence is represented by a vector. These vectors are used as input for LSTM neural networks (Hochreiter and Schmidhuber, 1997). Outputs of the LSTM network goes into a simple feed-forward network, and the output of this feed-forward network are weights which sum to 1. These weights are multiplied with the outputs of the LSTM network to give them weights. This mechanism lets the model learn which input from the protein sequence is crucial in classifying a gene into its correct antibiotic class. Finally, there is a softmax layer, the size of which is corresponds to the number of antibiotic classes we have. This layer assigns a probability to each class. The class that is assigned the highest probability becomes the prediction for a certain protein sequence.



(a) Precision score (b) Recall score (c) F_1 score

Figure 4.2: Precision, Recall, and F_1 score for Adam and pSGLD. Our deep learning models have a high accuracy for antibiotic resistance gene classification without using any kind of alignment, and just using the amino acids in protein sequences as features.



(a) Correct predictions

(b) Incorrect predictions

Figure 4.3: Probability assigned to the class predicted by the pSGLD, and ADAM models. (a) Probability assigned to a class when the models are correctly predicting. (b) Probability assigned to a class when the models are incorrectly predicting. Scale of the y-axis in (a) and (b) are different for ease in visibility.

Precision (Pr), Recall (Rc), and F_1 are defined as:

$$Pr = \frac{TP}{TP + FP};\ Rc = \frac{TP}{TP + FN};\ F_1 = 2 \times \frac{Pr \times Rc}{Pr + Rc}$$

Where TP: True Positives, FP: False Positives, FN: False Negatives.

We can see that overall the performance of both models are comparable. Adam has a 96% precision, 95% recall, and 96% F_1 score whereas pSGLD has a 96% precision, 96% recall, and 96% F_1 score.

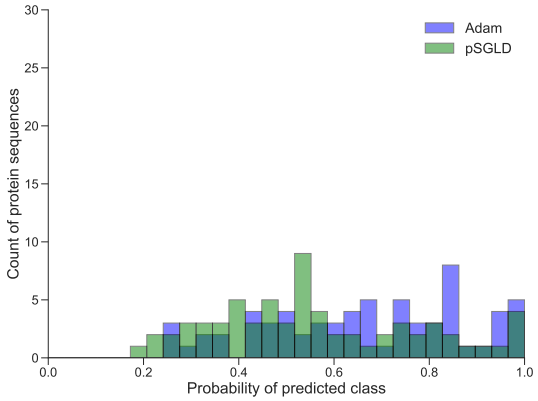
Figure 4.3 shows a histogram distribution of probability assigned to a class predicted for a protein sequence by both models. Figure 4.3a shows the distribution of probability of a predicted class when both models are correct. Figure 4.3b shows the distribution of probability of a predicted class when both models are incorrect. We can see that when the models are predicting correctly, the probability assigned to the correct class by both models are high. In contrast, when the models are incorrect, pSGLD tends to assign somewhat lower probability to the predicted class signalling its uncertainty.

Result on OoD data Next, we tested both models on OoD samples. For this we used three datasets - **(i)** The 19 classes we discarded out of the 34 antibiotic classes from the DeepARG dataset. We did not include protein sequences from these classes in training and testing our models. These 19 classes have a total of 67 protein sequences. **(ii)** 19,576 human genes collected from UNIPROT (Apweiler et al., 2004) that we can confidently assume do not confer antibiotic resistance. **(iii)** 480 bacteria genes that are involved in the metabolism pathway were curated from the KEGG (Kanehisa, 2002) database. These have very low chances of involvement in conferring antibiotic resistance.

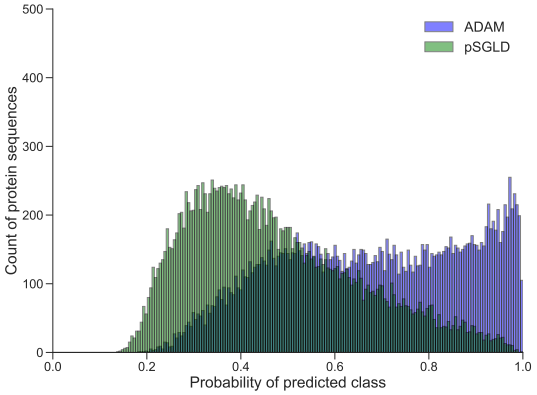
Before testing on these sequences, our expectation is that an ideal model trained to do multiclass classification on a set of classes should provide low probabilities for its prediction on sequences that belong to classes not included in this set. The model should also provide low probabilities for human genes and bacteria metabolism genes that do not confer antibiotic resistance.

Figure 4.4 shows a histogram distribution of probability assigned to the predicted class for both models on these sequences. We can see from the figure that in all three cases the probability distribution for pSGLD is centered around 0.5 or left whereas for ADAM the distribution is heavily right skewed. The ADAM trained model is still predicting these OoD sequences to be one of the 15 classes it was trained on with high confidence. In contrast, the pSGLD model is conveying its uncertainty over its predictions.

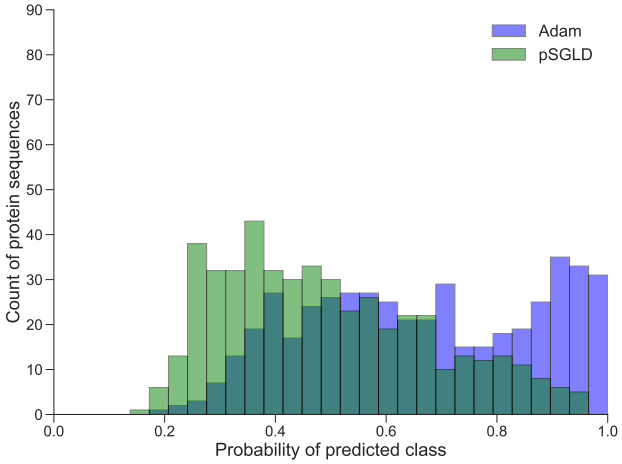
Next, we tested both models in a binary classification setting where our positive class includes the test set of the 15-class dataset from DeepARG. This test set has 1491 genes. For our negative dataset, we used the three OoD datasets mentioned previously. We tested both models, and measured their performance in terms of classifying the negative class from the positive class. The models were not trained to do this binary classification. They were still only trained to predict an antibiotic class from the 15 classes. But we are testing them to see how good they are when facing negative data just like in a real world scenario. Training to do a specific task, and then testing to detect OoD data is also used in computer vision to measure the performance of a system on identifying OoD data (Liang et al., 2017; DeVries and Taylor, 2018).



(a) Probability assigned to the predicted class for protein sequences from 19 OoD antibiotic classes



(b) Probability assigned to the predicted class for the 19,576 human genes that are not antibiotic resistance genes



(c) Probability assigned to the predicted class for the 480 Bacteria genes that are involved in metabolism pathways

Figure 4.4: Probabilities assigned to the predicted class by both pSGLD and ADAM trained models. The pSGLD trained model has left-skewed probability distribution on the OoD protein sequences while the ADAM trained model has right-skewed probability distribution implying high confidence in its predictions. Scale of the y-axis in (a), (b), and (c) are different for ease in visibility.

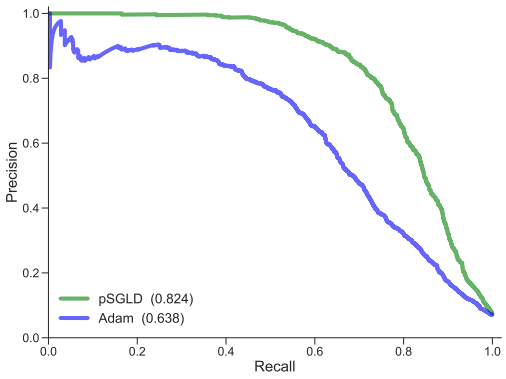
Figure 4.5 shows the Precision-Recall (PR) curves and Receiver Operating Characteristic (ROC) curves for the binary classification task. We can see that pSGLD model has PR and ROC curves that cover a greater area than the ADAM model when negative data are the human and bacteria genes. Specially, for human genes, performance of pSGLD is significantly better indicating pSGLD is more capable of detecting data it has not seen in training. pSGLD also performs better when negative data are bacteria genes not involved in antibiotic resistance. Performance of pSGLD and ADAM is almost similar when tested on the 67 genes from antibiotic classes that were not included in the training dataset.

4.6 Discussion

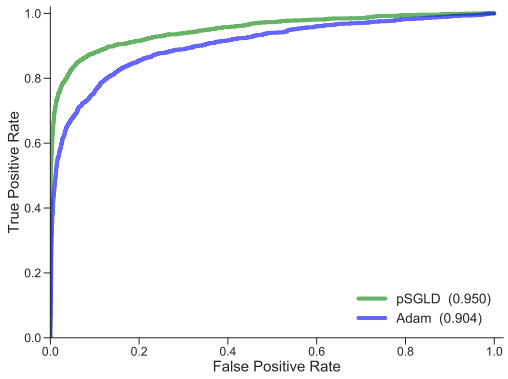
In summary, the contributions of this paper are as follows:

1. The self-attention based deep learning model provides extremely high accuracy in terms of precision, recall, and F_1 for antibiotic resistance gene classification. This is of crucial importance in many clinical settings. Additionally, the model only uses amino acids of protein sequences as its input removing any necessity for additional feature processing. In contrast, the DeepARG (Arango-Argoty et al., 2018) method first creates a similarity distance matrix by aligning sequences against CARD and ARDB datasets. This similarity matrix is used as features for the deep neural network.

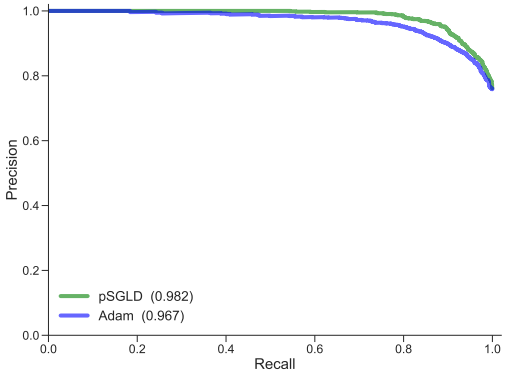
2. The self-attention based deep learning model trained with pSGLD provides reliable confidence scores for OoD gene detection. In the real world, we cannot expect that a model will see only antibiotic resistance genes that belong to one of the antibiotic classes that were included in its training. The model may encounter antibiotic resistance genes that confer resistance to antibiotics not included in its training set, or genes that do not have functionality of conferring antiobitc resistance at all. We need the model to provide low confidence scores when it encounters one of these two scenarios.



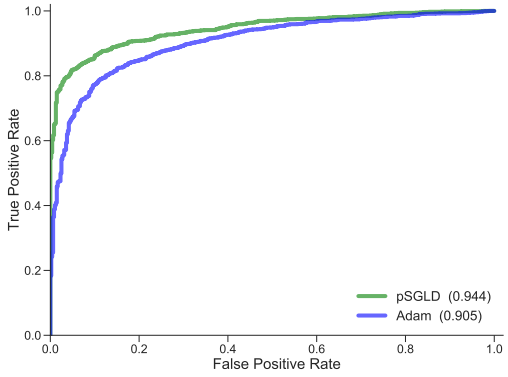
(a) PR curve where negative data is 19,576 Human genes



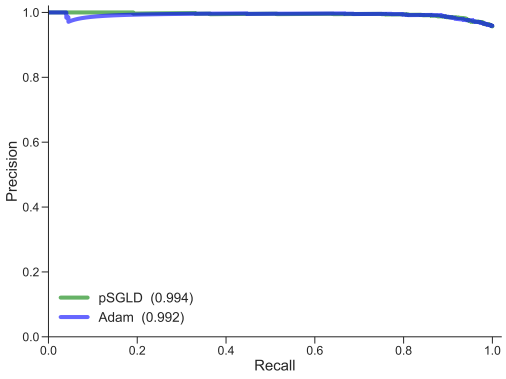
(b) ROC curve where negative data is 19,576 Human genes



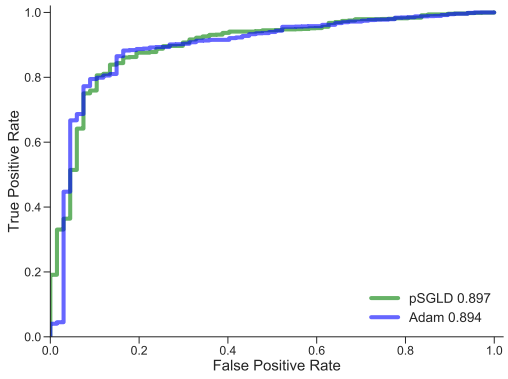
(a) PR curve where negative data is 480 Bacteria genes



(b) ROC curve where negative data is 480 Bacteria genes



(a) PR curve where negative data is 67 genes from 19 OoD classes



(b) ROC curve where negative data is 67 genes from 19 OoD classes

Figure 4.5: Precision-Recall (PR) curves and Receiver Operating Characteristic (ROC) curves for our binary classification task where for all cases positive data consists of the test set from our modified DeepARG dataset. These are 1491 genes that belong to one of the 15 antibiotic classes from the modified DeepARG dataset. For a and b, negative data is 19,576 Human genes that do not possess antibiotic resistance. For c and d, negative data is 480 Bacteria genes involved in the metabolism pathway, and do not possess antibiotic resistance. For e and f, negative data is 67 genes that are antibiotic resistant to 19 classes that were not included in the training and testing of our models.

3. *De-novo* antibiotic resistance gene classification overcomes the weakness of lots of false negatives resulting from alignment based methods such as BLAST (Altschul et al., 1990). This can be seen from the high recall score of our models. We refer to the DeepARG (Arango-Argoty et al., 2018) paper for a detailed expound on the weakness of predicting a high number of false negatives by BLAST.

4. With the self-attention based model, we present a viable use case of using entity embeddings (Guo and Berkhahn, 2016) as inputs for categorical variables. Amino acids are categorical variables, and usually represented in machine learning by one-hot vectors - vectors of size 20 where all indexes are zeros except the index of the amino acid we are representing which is 1. We show that, instead of using a one-hot vector, if we use a dense vector - a vector which has all non-zero real values - that is randomly initialized at the beginning, and then trained end-to-end inside the neural network, we can gain significant stride in terms of accuracy. These are known as entity embeddings in the literature. Also, there has been a trend on using word2vec type unsupervised learning to learn dense vectors for protein k-mers in biology (Asgari and Mofrad, 2015). Our results show that there is a need to consider end-to-end learning of dense vectors like in our model which can save precious time and computing that is needed for unsupervised learning.

5. DeepARG is based upon the Theano (Bergstra et al., 2010) deep learning framework which has been deprecated recently, and no longer being developed. This makes it difficult for new users to use the software. In contrast, our model is developed with the Pytorch (Paszke et al., 2017) deep learning framework which is being actively developed, and is used by a huge number of deep learning researchers. This will make it easier for new users to adopt the software which will be released soon.

In this study, we applied a self-attention based deep learning model to a multi-class classification task of predicting antibiotic class that a gene can confer resistance against. We trained our model with two different optimization algorithms, ADAM and pSGLD. The overall F_1 score for both

the ADAM and pSGLD trained models are 96%. Both models predict the antibiotic class of resistance genes with high accuracy. Comparison with the DeepARG model was not feasible as they use the CARD and ARDB protein sequences for generating features by aligning them against the UNIPROT protein sequences they curated. We could not find out, specifically, which protein sequences they used as their test data. Our model used all of the protein sequences from CARD, ARDB, and UNIPROT, and divided them into training, validation, and test set after shuffling them. Moreover, the test set performance reported in the DeepARG model suffers from data leakage. They selected protein sequences from UNIPROT by using an alignment threshold against the CARD and ARDB dataset when creating the complete dataset. Afterwards the same selected UNIPROT sequences were aligned against CARD and ARDB using Diamond (Buchfink et al., 2015), and bit scores from the result were used to make a similarity distance matrix. This matrix was used as features for a neural network to predict on test data that were previously selected from UNIPROT by alignment threshold against CARD and ARDB.

We also tested both of our models on three datasets of protein sequences that we know either belong to classes of antibiotic resistance that were not part of our training and testing, or are not antibiotic resistance associated. The ADAM trained model predicted them to be of the 15 classes with high confidence scores. In contrast, the pSGLD trained model provided predictions with a lower probability distribution for the proteins not associated with antibiotic resistance. We hypothesize that the Gaussian noise introduced in the pSGLD training scheme impedes the neural networks to completely collapse on the Maximum Likelihood solution. Therefore, a pSGLD trained model lets a discriminative model detect OoD data points, and provide lower probabilities in its predictions for them. This is an important property, especially when we consider the open world problem in biology where for any classification task it is hard to collect negative training samples to train the machine learning algorithm (Dessimoz et al., 2013). Future research might involve improving our neural network training methods so that the probability distribution of prediction for in-distribution and out-of-distribution data has much less overlap than what we could manage with

pSGLD. For that, we can take inspiration from the research on detection of adversarial instances (Szegedy et al., 2013) done by the computer vision community.

4.7 References

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. (1990). Basic local alignment search tool. *Journal of molecular biology*, 215(3):403–410.

Apweiler, R., Bairoch, A., Wu, C. H., Barker, W. C., Boeckmann, B., Ferro, S., Gasteiger, E., Huang, H., Lopez, R., Magrane, M., et al. (2004). Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 32(suppl 1):D115–D119.

Arango-Argoty, G., Garner, E., Pruden, A., Heath, L. S., Vikesland, P., and Zhang, L. (2018). Deeparg: a deep learning approach for predicting antibiotic resistance genes from metagenomic data. *Microbiome*, 6(1):23.

Asgari, E. and Mofrad, M. R. (2015). Continuous distributed representation of biological sequences for deep proteomics and genomics. *PloS one*, 10(11):e0141287.

Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., and Bengio, Y. (2010). Theano: A cpu and gpu math compiler in python. In *Proc. 9th Python in Science Conf*, volume 1.

Brown, E. D. and Wright, G. D. (2016). Antibacterial drug discovery in the resistance era. *Nature*, 529(7586):336–343.

Buchfink, B., Xie, C., and Huson, D. H. (2015). Fast and sensitive protein alignment using diamond. *Nature methods*, 12(1):59.

Bulatov, Y. (2011). Notmnist dataset. *Google (Books/OCR), Tech. Rep.[Online]. Available: http://yaroslavvb. blogspot. it/2011/09/notmnist-dataset. html*.

Choi, H. and Jang, E. (2018). Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*.

Consortium, U. et al. (2018). Uniprot: the universal protein knowledgebase. *Nucleic acids research*, 46(5):2699.

Davis, J. J., Boisvert, S., Brettin, T., Kenyon, R. W., Mao, C., Olson, R., Overbeek, R., Santerre, J., Shukla, M., Wattam, A. R., et al. (2016). Antimicrobial resistance prediction in patric and rast. *Scientific reports*, 6:27930.

Dessimoz, C., Škunca, N., and Thomas, P. D. (2013). Cafa and the open world of protein function predictions. *Trends in genetics : TIG*, 29(11):609–610.

DeVries, T. and Taylor, G. W. (2018). Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*.

Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

for Disease Control, C. and Prevention (2018). About Antimicrobial Resistance. https://www.cdc.gov/drugresistance/about.html.

Guo, C. and Berkhahn, F. (2016). Entity embeddings of categorical variables. *arXiv preprint arXiv:1604.06737*.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Jia, B., Raphenya, A. R., Alcock, B., Waglechner, N., Guo, P., Tsang, K. K., Lago, B. A., Dave, B. M., Pereira, S., Sharma, A. N., et al. (2016). Card 2017: expansion and model-centric curation of the comprehensive antibiotic resistance database. *Nucleic acids research*, page gkw1004.

Kanehisa, M. (2002). The kegg database. In *In SilicoSimulation of Biological Processes: Novartis Foundation Symposium 247*, volume 247, pages 91–103. Wiley Online Library.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Kleinheinz, K. A., Joensen, K. G., and Larsen, M. V. (2014). Applying the resfinder and virulencefinder web-services for easy identification of acquired antibiotic resistance and e. coli virulence genes in bacteriophage and prophage nucleotide sequences. *Bacteriophage*, 4(2):e27943.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *AT&T Labs [Online]. Available: http://yann. lecun. com/exdb/mnist*, 2.

Li, C., Chen, C., Carlson, D. E., and Carin, L. (2016). Preconditioned stochastic gradient langevin dynamics for deep neural networks.

Liang, S., Li, Y., and Srikant, R. (2017). Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*.

Lin, Z., Feng, M., Santos, C. N. d., Yu, M., Xiang, B., Zhou, B., and Bengio, Y. (2017). A structured self-attentive sentence embedding. *arXiv:1703.03130 [cs]*. arXiv: 1703.03130.

Liu, B. and Pop, M. (2008). Ardbantibiotic resistance genes database. *Nucleic acids research*, 37(suppl_1):D443–D447.

Neu, H. C. (1992). The crisis in antibiotic resistance. *Science*, 257(5073):1064–1073.

Organization, W. H. (2014). Antimicrobial resistance: global report on surveillance 2014. http://www.who.int/drugresistance/documents/surveillancereport/en/.

Pal, C., Bengtsson-Palme, J., Kristiansson, E., and Larsson, D. J. (2016). The structure and diversity of human, animal and environmental resistomes. *Microbiome*, 4(1):54.

Palacci, H. and Hess, H. (2018). Scalable natural gradient langevin dynamics in practice. *arXiv preprint arXiv:1806.02855*.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. (2017). Automatic differentiation in pytorch.

Perry, J. A. A., Westman, E. L. L., and Wright, G. D. (2014). The antibiotic resistome: what's new? *Current opinion in microbiology*, 21:45–50.

Robbins, H. and Monro, S. (1985). A stochastic approximation method. In *Herbert Robbins Selected Papers*, pages 102–109. Springer.

Robert, C. and Casella, G. (2004). Monte carlo statistical methods springer-verlag. *New York*.

Roberts, G. O. and Stramer, O. (2002). Langevin diffusions and metropolis-hastings algorithms. *Methodology and computing in applied probability*, 4(4):337–357.

Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Welling, M. and Teh, Y. W. (2011). Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688.

CHAPTER 5. DISCUSSION AND FUTURE DIRECTIONS

5.1 Discussion

In this dissertation, I tackled the problem of antibiotic resistance using two broad approaches: looking for novel natural sources of antibiotics, and making antibiotic resistance gene classification more efficient. In Chapter 2, I raise the issue of the urgent necessity of novel antibiotics. I showed that bacteriocins, peptide-based antimicrobials, can be a natural source of novel antibiotics. Because of their narrow killing spectrum, potential antibiotics generated from bacteriocins can also be effective against their target microbes for longer without them gaining resistance. However, since bacteriocins are sequentially diverse, alignment based algorithms such as BLAST or HMMER are limited in detecting novel bacteriocins from protein sequences.

To overcome this limitation, I proposed a machine learning based alignment free approach to predict bacteriocins from protein sequences. I collected 346 bacteriocin protein sequences, and developed a negative dataset of 346 non-bacteriocin protein sequences. As this is a relatively small labeled dataset, I decided to augment the model with a transfer learning approach. Vector representations of 3-mers of amino acids were learnt using the large Uniprot TrEMBL dataset of unlabeled protein sequences. For this purpose, the word2vec algorithm from natural language processing was used which takes into account the neighboring words of a word to learn its representation. This method takes the neighboring context of a word, and embeds it into the representation of the word. In this case, for each 3-mer, the word2vec algorithm took into account its neighboring 3-mers in all protein sequences to learn its vector representation.

After learning these representations, we represented each protein sequence of our original small labeled dataset as the summation of all overlapping 3-mers in that sequence. Then we tried various supervised algorithms such as RNN, SVM, and others with these representations to measure their performance in terms of classifying a positive bacteriocin from a negative one. We also compared

their performances against alignment based approaches such as BLAST and HMMER. Additionally, we measured the performances of the above mentioned supervised learning algorithms with a simple trigram representation of the protein sequences. From all the comparisons in terms of F_1 score and precision-recall curve, we show that a word2vec representation with either RNN or SVM significantly improves performance of the classifier compared to all the other methods and representations.

Consequently, we used the best performing method, word2vec representation with RNN on all protein sequences from *Lactobacillus* which resulted in the identification of 119 putative bacteriocins which could not be found by using BLAST. By using a transfer learning approach of learning representations of 3-mers from unlabeled protein sequences, and using these representations in our labeled dataset, we transferred knowledge of the surroundings of individual 3-mers into our small labeled dataset. This added knowledge enhanced the performance of the downstream classifier. By showing an effective use case in terms of finding novel bacteriocins, we also show the efficacy of word embedding representations to find functionally similar but non-homologous protein sequences which cannot be found by homology search methods such as BLAST or HMMER.

In Chapter 3, we denote the critical need of finding the antibiotic class genes from a clinical or environmental sample can confer resistance against. This can be crucial to gain guidance for patient treatment or for antibiotic resistance surveillance. Efficient computational tools that can provide such critical information can greatly complement traditional experimental culture based techniques by reducing wait time as well as providing added certainty. For this purpose, we created a benchmark dataset, COALA, that comprises protein sequences and the antibiotic class label they confer resistance to, from 15 antibiotic resistance databases. We also propose a transfer learning approach with deep neural networks, TRAC, and compare it with six other methods for antibiotic resistance gene classification. All the methods are tested on COALA40 and COALA70, two datasets that are trimmed down from the original COALA dataset by removing similar sequences with a CD-HIT threshold of 0.4 and 0.7 respectively.

Our proposed method, TRAC, consists of three steps with a deep recurrent neural network. In the first step, we train the deep recurrent neural network to predict the next amino acid in a protein sequence. For this purpose, we train the model with all bacteria protein sequences from the UniRef50 dataset. This is inspired by the language modeling technique used in natural language processing. A language model learns a probability distribution over all possible sentences in a language by trying to predict the next word in a sentence during training. In our case, the language model tries to learn the probability distribution over all possible protein sequences by trying to predict the next amino acid in a protein sequence. In the second step, we take the trained language model over UniRef50, and train it again on our COALA40 and COALA70 dataset protein sequences. This helps the neural network learn the probability distribution of amino acids relevant to our antibiotic resistance classification task. In the final step, we add a final layer on the trained neural network so that it can predict the antibiotic resistance class from a protein sequence.

We compare TRAC with 4 currently used tools in the field, CARD-RGI, NCBI-AMRFinder, SARGFAM, and DeepARG. We also develop another deep learning model ourselves that is only trained on the labeled COALA40 and COALA70 dataset without any previous language modeling. This is to find out whether the transfer learning approach of pretraining enhances classification performance. We also develop a machine learning approach, a random forest classifier with a simple trigram representation of protein sequences to compare performances against the deep learning approaches. We performed a 10-fold nested cross validation on both COALA40 and COALA70 with all the methods. In terms of mean accuracy, TRAC performs significantly better than all other methods. Performance of the deep learning model from scratch clearly shows that language model pretraining helps the neural network augment its accuracy. We hypothesize that learning a probability distribution over all bacteria genes, and then over antibiotic resistance protein sequences helps the neural network recognize essential signals for downstream antibiotic resistance classification task. Surprisingly, a random forest model with basic trigram representation of protein sequences perform similarly to a deep learning model without any pretraining. Among the 4 methods currently used in literature, CARD-RGI performs best. We hypothesize that CARD-RGI

does better because it uses other models together with a homolog model such as a protein variant model. Methods such as SARGFAM and NCBI-AMRFinder do worse as their methods are limited to homology search methods such as BLAST and HMMER. Surprisingly, DeepARG, a deep learning based model has the poorest performance, which might be due to the case that their training and testing dataset contains significant overlap in terms of sequence similarity. As a result, the DeepARG model might be significantly overfitting showing a good performance in the test set used in their work but performing worse in wild.

In Chapter 2 and 3, we have used two different types of transfer learning approaches. In Chapter 2, we used the word2vec algorithm to learn static vector representations of 3-mers from unlabeled protein sequences that takes into account their surrounding 3-mers. We used these representations to represent protein sequences in the small labeled dataset of positive and negative bacteriocins. In contrast, in Chapter 3, we used a language model trained on unlabeled protein sequences, and finally added a classification layer over the language model neural network to predict antibiotic resistance class. In natural language processing, the language modeling type deep transfer learning has almost completely replaced the word2vec type shallow transfer learning because of its superior performance. As such, we hypothesize that, a language model pretrained with unlabeled protein sequences might increase the prediction performance of bacteriocins which might be an avenue of future research. But in both chapters we show that transfer learning approaches significantly outperform alignment based approaches such as BLAST and HMMER. This further strengthens the argument that deep learning based approaches can find functionally similar but non-homologous sequences more efficiently. This should excite us as there are many areas in computational biology where biologists still rely on alignment based approaches to find functionally similar genes. Transfer learning based approaches might radically change the landscape of functional genomics by finding novel genes of both known and unknown functions.

In Chapter 4, we try to approach the problem of antibiotic resistance gene classification from an orthogonal way to our approach in Chapter 3. While in Chapter 3, we try to maximize the accuracy of the model to predict a gene's antibiotic resistance class, in Chapter 4, we try to incorporate the

model's ability to say 'I don't know' when it encounters a gene that does not possess the functionality of conferring antibiotic resistance or a gene that is not from bacteria at all. The model does this by producing a low probability with its prediction when it encounters such an instance. This is critical because we cannot always be sure of what input a model will get when deployed in the wild. To this end, we train our neural network model with an optimization algorithm called pSGLD. This optimization algorithm introduces gaussian noise into the backpropagation algorithm. Backpropagation algorithm is what allows a neural network to correct its mistakes which it knows in turn by comparing with the true label. By introducing this noise, the neural network becomes more conservative in its prediction probabilities, and even more so when encountering data that is out of distribution of the data it was trained on. We compare the performances of our neural network by training the same model with two different optimization algorithms, pSGLD, and the popular ADAM algorithm that is available in every deep learning framework. Their performances in terms of precision, recall and F_1 score are comparable. But when tested with out of distribution data such as bacteria genes that do not possess antibiotic resistance functionality or genes from human, the pSGLD trained model outputs a low prediction probability compared to an ADAM trained model. The pSGLD trained neural network is much more efficient in relaying its uncertainty when it gets data from a distribution it has not seen in training. In contrast, the ADAM trained model still provides prediction probability for such data that is not much different to its prediction probabilities for antibiotic resistance genes.

By integrating the work in Chapter 4 with the work in Chapter 3, we can get a classifier that not only performs well in terms of accurately predicting antibiotic resistance class but also conveys its uncertainty when it detects non-antibiotic resistance genes. This can be an exciting avenue for future research. Combining transfer learning and reliable uncertainty estimation is in fact the holy grail of many critical applications such as medical prognosis.

5.2 Future directions

The work in this dissertation is the first time transfer learning has been introduced to tackle problems in the antibiotic resistance domain in my knowledge. The potential of different kinds of transfer learning is still waiting to be explored. To make the bacteriocin hunting process more efficient, there is the potential of semi-supervised learning with graph neural networks (Kipf and Welling, 2016) which are neural networks that use graph structured data. As we saw from Chapter 2, we have a small labeled set of positive bacteriocins. Can we use gene regulatory networks from bacteria, and turn this problem of finding bacteriocins into a network problem? Can we use the vast unlabeled and small labeled data to perform semi-supervised learning with graph neural networks? Another exciting and challenging task would be to predict secondary structures of bacteriocins from their protein sequences. We know very little about the secondary and tertiary structures of bacteriocins (Van Belkum et al., 2011; Daw and Falkiner, 1996; Ennahar et al., 2000; Aleksandrzak-Piekarczyk,). Learning about their secondary and tertiary structures might reveal potential insights that would help us find them more efficiently, and even produce novel synthetic antibiotics. Transfer learning can take advantage of the great number of labeled structure data, and the smaller number of structure data of bacterial antimicrobial peptides available from the protein data bank (Burley et al., 2018). Especially since deep neural networks are currently performing the best for the protein structure prediction task (AlQuraishi, 2019), there has never been a more exciting time for structure prediction problems where there are few labeled data.

For the antibiotic resistance classification task, while TRAC performs much better than current methods in the area, there is still serious room for improved performance as can be seen from its accuracy. Can we use all UniRef50 sequences instead of just bacteria sequences for pretraining, and find out if that improves downstream classification performance in a notable way? We used a deep recurrent neural network based architecture for the language modeling task. In natural language processing, the language modeling task has now been taken over by attention based models such as Transformers (Vaswani et al., 2017). Will the use of Transformers for language modeling in protein sequences take our classification performance to the next level? These are some exciting future

directions for this task that can be undertaken. For the uncertainty estimation task, there is the potential of novel techniques introduced in the field of deep learning for out of distribution data detection (Liang et al., 2017; Frosst et al., 2019; DeVries and Taylor, 2018). The task of integrating properly calibrated uncertainty estimation with superior accuracy on the COALA40 and COALA70 datasets can be of immediate undertaking.

5.3 References

Aleksandrzak-Piekarczyk, T. Struktura, funkcjonowanie i regulacja systemów metabolizmu cukrów u bakterii fermentacji mlekowej.

AlQuraishi, M. (2019). Alphafold at casp13. *Bioinformatics*.

Burley, S. K., Berman, H. M., Bhikadiya, C., Bi, C., Chen, L., Di Costanzo, L., Christie, C., Dalenberg, K., Duarte, J. M., Dutta, S., et al. (2018). Rcsb protein data bank: biological macromolecular structures enabling research and education in fundamental biology, biomedicine, biotechnology and energy. *Nucleic acids research*, 47(D1):D464–D474.

Daw, M. A. and Falkiner, F. R. (1996). Bacteriocins: nature, function and structure. *Micron*, 27(6):467–479.

DeVries, T. and Taylor, G. W. (2018). Learning confidence for out-of-distribution detection in neural networks. *arXiv preprint arXiv:1802.04865*.

Ennahar, S., Sashihara, T., Sonomoto, K., and Ishizaki, A. (2000). Class iia bacteriocins: biosynthesis, structure and activity. *FEMS microbiology reviews*, 24(1):85–106.

Frosst, N., Papernot, N., and Hinton, G. (2019). Analyzing and improving representations with the soft nearest neighbor loss. *arXiv preprint arXiv:1902.01889*.

Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Liang, S., Li, Y., and Srikant, R. (2017). Enhancing the reliability of out-of-distribution image detection in neural networks. *arXiv preprint arXiv:1706.02690*.

Van Belkum, M. J., Martin-Visscher, L. A., and Vederas, J. C. (2011). Structure and genetics of circular bacteriocins. *Trends in microbiology*, 19(8):411–418.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.