Predicting drug resistance in *M. tuberculosis* using a Long-term Recurrent Convolutional Network

by

Amir Hosein Safari

B.Sc., Sharif University of Technology, 2019

Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

in the
School of Computing Science
Faculty of Applied Sciences

SIMON FRASER UNIVERSITY
Summer 2021

Declaration of Committee

Name:	**Amir Hosein Safari**
Degree:	**Master of Science**
Thesis title:	**Predicting drug resistance in *M. tuberculosis* using a Long-term Recurrent Convolutional Network**
Committee:	**Chair:** Mo Chen Assistant Professor, Computing Science
	Maxwell Libbrecht Supervisor Assistant Professor, Computing Science
	Leonid Chindelevitch Committee Member Associate Professor, Computing Science
	Angelica Lim Examiner Assistant Professor, Computing Science

Abstract

Motivation: Drug resistance in Mycobacterium tuberculosis (MTB) is a growing threat to human health worldwide. One way to mitigate the risk of drug resistance is to enable clinicians to prescribe the right antibiotic drugs to each patient through methods that predict drug resistance in MTB using whole-genome sequencing (WGS) data. Existing machine learning methods for this task typically convert the WGS data from a given bacterial isolate into features corresponding to single-nucleotide polymorphisms (SNPs) or short sequence segments of a fixed length K (K-mers). Here, we introduce a gene burden-based method for predicting drug resistance in TB. We define one numerical feature per gene corresponding to the number of mutations in that gene in a given isolate. This representation greatly reduces the number of model parameters. We further propose a model architecture that considers both gene order and locality structure through a Long-term Recurrent Convolutional Network (LRCN) architecture, which combines convolutional and recurrent layers.

Results: We find that using these strategies yields a substantial, statistically significant improvement over state-of-the-art methods on a large dataset of *M. tuberculosis* isolates, and suggest that this improvement is driven by our method's ability to account for the order of the genes in the genome and their organization into operons.

Availability: The implementations of our feature preprocessing pipeline[1] and our LRCN model[2] are publicly available, as is our complete dataset[3].

Keywords: infectious disease, deep learning, antimicrobial resistance, tuberculosis, next-generation sequencing

[1] https://github.com/AmirHoseinSafari/Genotype-collector-and-SNP-dataset-creator

[2] https://github.com/AmirHoseinSafari/LRCN-drug-resistance

[3] https://github.com/AmirHoseinSafari/M.tuberculosis-dataset-for-drug-resistant

Acknowledgements

This thesis would not have been possible without the support of several thoughtful and generous individuals. My advisors, Professor Maxwell W Libbrecht and Professor Leonid Chindelevitch because of their great support and help from the beginning of my MSc study until the very end of it.

Also I would like to thank my colleagues whom helped me in this project. Especially Mr. Hooman Zabeti, Mrs. Nafiseh Sedaghat, and Mr. Alpha Forna.

My deepest gratitude goes to my family for their unconditional love and support throughout my life and also my friends.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Drug resistance is the phenomenon whereby an infectious organism (also known as a pathogen) develops resistance to the drugs that are commonly used in its treatment [42]. In this article, our focus is on *Mycobacterium tuberculosis* (MTB), the etiological agent of tuberculosis (TB). It is the deadliest infectious disease today and is responsible for almost 2 million deaths every year among 10 million new cases [43]. The challenge of drug-resistant TB is not only a concern for low and middle-income countries, but also high-income countries [32]. The importance of drug-resistant TB—and other drug-resistant pathogens—is due to the fact that, without novel antimicrobial drugs, the total deaths due to drug resistance may exceed 10 million people a year by 2050, which is higher than the current annual mortality due to cancer [28].

One way to mitigate the risk of drug resistance is to carry out a drug susceptibility test (DST) by growing the bacterial isolate in the presence of different drugs and prescribing a regimen consisting of drugs the isolate is susceptible to. However, this approach is time-consuming and labour-intensive, so treatment is typically started before its results become available, potentially leading to poor outcomes.

The use of whole-genome sequencing (WGS) data makes it possible to identify drug resistance in hours rather than days. Prior methods for identifying drug resistance from WGS data can be divided into two categories. The first category consists of catalogue methods, which involve testing the WGS data of an isolate for the presence of known mutations associated with drug resistance. These mutations are primarily single nucleotide polymorphisms (SNPs), though they can also be insertions or deletions (indels) [2]. An isolate is then predicted to be resistant if it has one or more such mutations [38, 5, 1, 16, 13]. However, this approach often has poor predictive accuracy [34], especially for situations involving novel drug resistance mechanisms or resistance to untested or rarely-used drugs [17].

The second category consists of machine learning (ML) methods, which aim to to predict drug resistance by using models trained directly on paired WGS and DST data [44, 3, 10, 7, 20, 19, 46]. The current state-of-the-art models include the wide-n-deep neural network

(WnD) [3], DeepAMR [44], KOVER [10], and gradient-boosted trees (GBTs) [7, 20]. While existing machine learning methods achieve better accuracy than catalogue methods, there remains much room for improvement, especially for the less commonly-used second and third-line drugs, and such an improvement in accuracy holds the promise of positively affecting clinical outcomes. Also, the lack of interpretability is a considerable disadvantage of most existing machine learning methods. We refer the reader to Appendix Section A for an extensive review of existing ML methods for drug resistance prediction.

Existing ML methods typically convert their input WGS data into features that correspond to SNPs or short sequence segments of a fixed length K (K-mers). These representations do not necessarily take advantage of the organization of genome sequences into genes, despite the fact that genes are the unit of structure supporting the functional changes responsible for drug resistance.

In the first Chapter of this article, we propose a data pipeline in order to obtain the SNP-based and gene-based datasets from the raw data. This pipeline gets the name of isolates as its input and it then 1) Downloads the WGS data of these isolates. 2) By using a combination of GATK and SAMtools, it obtains the SNPs of each isolate and then generates the SNP-based data, a binary matrice in which each row representing an isolate and each column representing an SNP. 3) By using the SNP-based data and TB genes information, it creates the gene-based data, which represents the number of mutations in each gene of a given isolate.

In Chapter 2, we introduce a gene-centric method for predicting drug resistance in TB. We define one feature per gene which represents the number of mutations in that gene in a given isolate. This representation greatly decreases the number of features—and therefore model parameters—relative to a SNP-based representation. To our knowledge, while gene-based statistical tests—known as gene burden tests—are sometimes used for microbial genome-wide association studies [8, 12], and have been used to represent rare variants in a previous ML method [3], this work is the first systematic use of such gene burden features for drug resistance prediction using ML methods. In contrast to previous practice, where only the mutations leading to a specific type of protein change such as deleterious or non-synonymous mutations contribute to the gene's burden, our features count all the mutations found in a gene.

We further propose a model that accounts for both the order and the locality structure of genes through a Long-term Recurrent Convolutional Network (LRCN) architecture, which combines convolutional and recurrent layers. LRCNs have recently been successfully used in other fields such as computer vision [9, 25] but, to our knowledge, this is the first use of an LRCN in computational biology for a task that is not directly related to image processing. We also introduce a multi-task approach for this problem, where we train a single model to jointly predict resistance to twelve drugs [3, 45].

Remarkably, we find that using these strategies yields a substantial improvement over the state-of-the-art tools. This improvement is statistically significant and consistent across many drugs and settings, and requires both elements of our model, namely, the gene burden-based features and the LRCN model architecture. We verify via permutation testing that the order of genes and their organization into operons drive the model's performance. Based on these results, we expect that this gene burden-based method may prove useful for other genotype-to-phenotype prediction problems.

Moreover in the last Chapter, in order to make the LRCN model interpretable and see which genes are more responsible for the drug resistance phenotype, we combined our LRCN model with the interpretable model-agnostic explanations (LIME) method. Therefore, we obtained the importance score for each feature in our gene-based data and we observed that there is a high correlation between the genes that we know are responsible for drug resistance phenotype from the Biology perspective and the genes that has a high importance score in LRCN predictions.

Chapter 2

Data pipeline

To train and evaluate our method, we used the Pathosystems Resource Integration Center (PATRIC) [41] and the Relational Sequencing TB Data Platform (ReSeqTB) [37] datasets to collect 7,845 isolates, together with their resistance/susceptible status (labels) for twelve drugs. These include five first-line drugs—isoniazid, rifampicin, ethambutol, pyrazinamide, and streptomycin; three injectable second-line drugs— amikacin, capreomycin, and kanamycin; three fluoroquinolones— ciprofloxacin, moxifloxacin, and ofloxacin; and one less commonly used second-line drug, ethionamide (Table 2.1) [26, 7]. The short reads containing the whole-genome sequences of these 7,845 isolates were downloaded from the European Nucleotide Archive [22] and the Sequence Read Archive [21]. The accession numbers which we used in our dataset were ERP[000192, 006989, 288008667, 010209, 013054, 000520], PRJEB[10385, 10950, 14199, 2358, 2794, 5162, 9680], PRJNA[183624, 235615, 296471], and SRP[018402, 051584, 061066]. This is the same dataset we used in a previous study, whose focus was on the development of interpretable drug resistance prediction models [46].

We used an additional, independent, dataset to test how well the results generalize. We downloaded the SNP-based feature matrix of a dataset from the British Columbia Centre for Disease Control (BCCDC), used in another publication by our group [14]; we describe this dataset in more detail in Table 2.2. This dataset contains over 1,138 TB isolates with their labels for the five first-line drugs: isoniazid, rifampicin, ethambutol, pyrazinamide, and streptomycin. We used this dataset for the results shown in Figure 3.5, and the PATRIC and ReSeqTB datasets for the remainder of the experiments.

2.1 Variant calling

To obtain the SNP information from the raw reads of the PATRIC and ReSeqTB datasets, we used a standard protocol [7, 6]. We first mapped the raw sequence data to the reference genome, *Mycobacterium tuberculosis H37Rv*, using `bwa-mem` [23], then called the variants (SNPs, insertions, and deletions) for each isolate using two different established pipelines, SAMtools [24] and GATK [31]. To make the calls more robust, we used the intersection

Drug	Num of labelled isolates	Num of resistant isolates
Streptomycin	5,125	2,104 (41%)
Rifampicin	7,715	2,968 (38%)
Pyrazinamide	3,858	754 (20%)
Ofloxacin	2,911	800 (27%)
Moxifloxacin	961	129 (13%)
Kanamycin	2,436	697 (27%)
Isoniazid	7,734	3,445 (45%)
Ethionamide	1,516	498 (33%)
Ethambutol	6,096	1,407 (23%)
Ciprofloxacin	443	37 (8%)
Capreomycin	1,991	552 (28%)
Amikacin	2,033	573 (28%)

Table 2.1: The number of isolates and the label distribution in our data.

of variant calls between the two tools; fewer than 4% of the variants were removed in this step. These SNP-based features were then represented as a binary matrix of 7,845 isolates (in rows) by 742,620 variants (in columns). The BCCDC dataset was processed into variant calls [14] before we acquired it, so we skipped this step for that dataset.

We created gene burden features by identifying, for a given isolate, the number of variants in each gene. We acquired boundaries for each known TB gene from MycoBrowser [18]. Of the 4,187 known TB genes, we found that 3,960 had a variant in at least one isolate in our dataset. We represent these gene burdens of the PATRIC and ReSeqTB datasets as an integer 7,485 by 3,960 matrix, in which each entry indicates the number of mutations in a given gene within a given isolate.

The advantage of using gene burden features instead of SNP-based features is that gene burden data drastically reduces the number of features, mitigating the "curse of dimensionality". When the number of available isolates is small relative to the number of SNPs, using the gene burden data may lead to more accurate models and reduce overfitting. The gene burden-based method may lose this advantage in situations when much larger sample sizes are available.

Drug	Num of labelled isolates	Num of resistant isolates
Streptomycin	1,136	87 (7.65%)
Rifampicin	1,138	26 (2.28%)
Pyrazinamide	134	13 (9.7%)
Isoniazid	1,138	157 (13.79%)
Ethambutol	1,138	22 (1.96%)

Table 2.2: Summary of the number of isolates and the label distribution in BCCDC data.

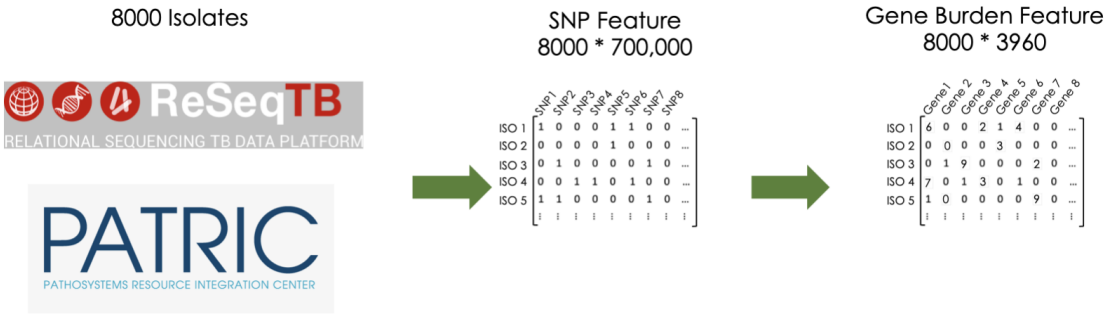


Figure 2.1: We proposed a data pipeline that at first we collected the WGS data of 8000 isolates from PATRIC and RESQTB with their resistance/susceptible status (labels) for twelve drugs. Then we obtained the SNP feature matrix and then the gene-burden feature matrix as we described in this section.

Chapter 3

LRCN model

3.1 Materials and Methods

3.1.1 LRCN model

Our method is based on a combination of long short-term memory (LSTM) [15] layers and convolutional neural network (CNN) [35] layers, an architecture known as a Long-term Recurrent Convolutional Network (LRCN) [9]. An LSTM is an artificial recurrent neural network that can learn long-distance dependencies by using feedback connections, which is appropriate for our situation given the multifactorial nature of the drug resistance phenotype [40]. More precisely, it is known that drug resistance in *M. tuberculosis* is mediated by compensatory mutations, which help reduce the fitness cost of drug-resistance causing mutations, but may occur in a different gene.

A CNN is a feed-forward neural network designed for processing structured arrays of data, especially data whose linear order is important. We elaborate on this further in Section 3.2.2, where we show that the linear order is important for our gene burden-based features. The combination of a CNN with an LSTM enables the model to take into account both the linear order as well as the local structure of the genes in a genome.

The architecture of our model is as follows. The input for each isolate is encoded as an integer vector $I = \{g_1, ..., g_m\}$, where m is the number of genes used (here, $m = 3,960$) and g_i is the number of SNPs the isolate has in the i-th gene. The output for this isolate is a binary vector $O = \{d_1, ..., d_n\}$ where n is the number of drugs (here, $n = 12$), and d_j is the isolate's predicted status for the j-th drug. This input is first processed by $L_{conv} = 2$ convolutional layers (`keras.layers.Conv1D` and `keras.layers.MaxPooling1D`) with specific filter (f_i), kernel (k_i), and pool (p_i) sizes, where i represents the layer number. These layers use the rectified linear unit (ReLU) activation function and the "same" padding (padding evenly to the left/right of the input such that the output has the same width dimension as the input). The output of the last CNN layer is processed by $L_{lstm} = 2$ LSTM layers (`keras.layers.LSTM`), with a specified number of nodes ($lstm_i$). They use the hyperbolic tangent activation function and a recurrent dropout with a rate of $p_{rec} = 0.3$. At the end,

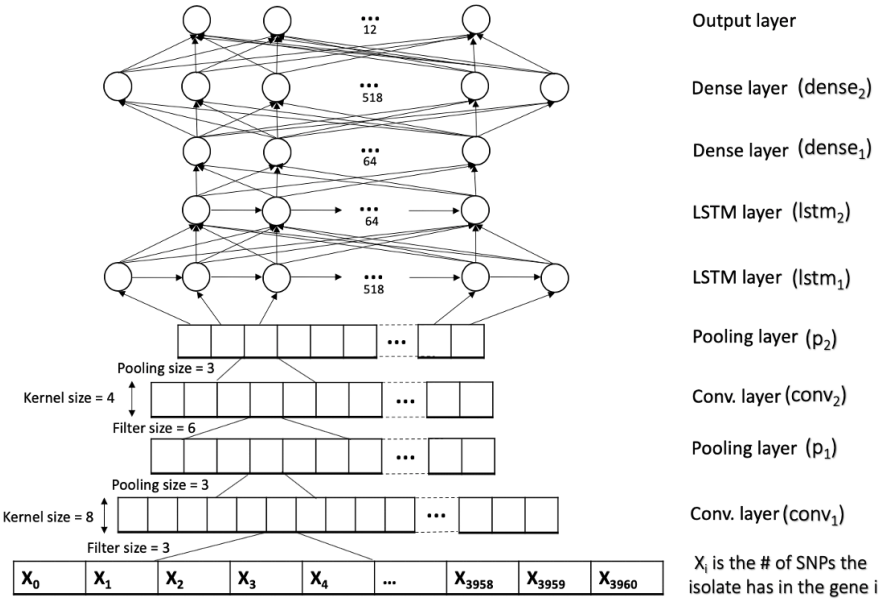


Figure 3.1: The architecture of our LRCN network.

the output of the LSTM feeds into the dense layers (`keras.layers.Dense` with a specified number of nodes ($dense_i$), and the last layer generates the output vector. Between every pair of layers, we use a dropout (`keras.layers.Dropout`) with a rate of $p = 0.1$ to reduce overfitting. A schematic representation of this architecture is shown in Figure 3.1.

The parameters of the optimized model (for the non-nested cross-validation approach in section 3.1.2) are as follows. For the CNN layers, the parameters are $(f_1, k_1, p_1) = (8, 3, 3)$ and $(f_2, k_2, p_2) = (4, 6, 4)$. The LSTM layers have $lstm_1 = 518$ and $lstm_2 = 64$ nodes, respectively. Finally, the dense layers have $dense_1 = 64$ and $dense_2 = 518$ nodes, respectively, and the last dense layer has $n = 12$ nodes to generate the output vector (Figure 3.1). We use the Adam optimizer with a learning rate of $\eta = 0.01$ to train the model. The parameters chosen by the nested cross-validation approach (Section 3.1.2) are listed in Appendix Section C.

We use a multi-task model, which predicts an isolate's resistance status for all 12 drugs in a single network. This approach's advantage is that if two drugs have a shared structure, either through highly correlated resistance status vectors due to their joint use in clinical regimens or through common resistance mechanisms, then the patterns learned from one drug can compensate for a smaller amount of training data for the other drug. The pairwise correlation of the status vectors for each pair of drugs is shown in Figure 3.2.

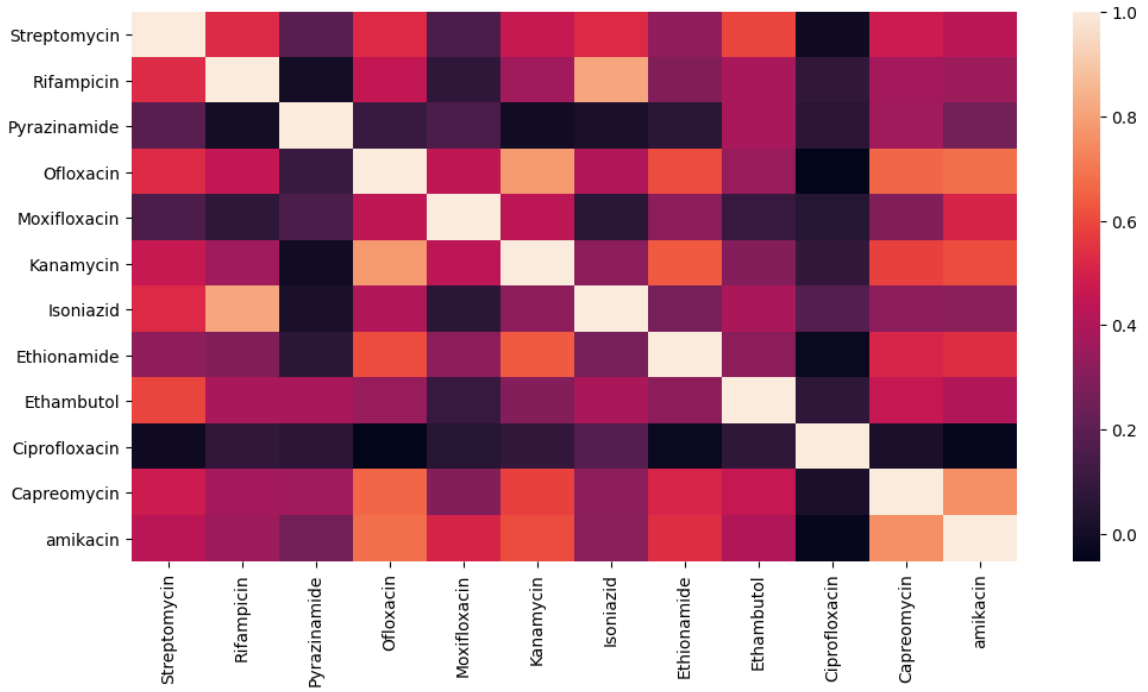


Figure 3.2: The pairwise correlation of the status vectors for each pair of drugs.

The challenge of using the multi-task model in our dataset is that many isolates lack labels for some of the drugs. For this reason we replace the usual loss function with a masked loss function. The masked loss function ignores the missing labels in calculating the loss, and therefore, these missing labels do not affect the network weights. We use the binary cross-entropy as the loss function. Specifically, we use the loss function

$$\text{Loss } = \sum_{i=1}^{I} \sum_{d=1}^{D} \mathbb{1}(X_{i,d} \text{ is available}) H(X_{i,d}, Y_{i,d}) \tag{3.1}$$

where I and D are the numbers of isolates and drugs respectively, $X_{i,d}$ and $Y_{i,d}$ are the true and predicted resistance values, H is the binary cross-entropy function, and $\mathbb{1}$ is the indicator function.

3.1.2 Train-Validation-Test split

To evaluate our method, we split our data into training, validation, and test sets. We use two different approaches for this purpose, as we detail below.

To obtain the results shown in Figure 3.4 and 3.5, we used a nested cross-validation approach. We split the data into 10 equal parts, and trained the model 10 times, using 8 folds for training, 1 fold for validation, and 1 for testing each time. In each run, we used Bayesian Optimization (see Section 3.1.4) to maximize the Area Under The Receiver Operating Characteristic Curve (AUC-ROC) on the validation set, and after tuning the

hyperparameters, we tested the best model on the test set. At the end of this process, we ended up with 10 different models for each method, from which we determined the mean and standard error of the AUC-ROC.

Because the approach above is computationally expensive, we used regular (non-nested) cross-validation for figures other than Figure 3.4 and 3.5. We used 10% of the data as the testing set. We chose hyperparameters via 10-fold cross-validation on the remaining dataset by selecting the model that achieved the highest mean AUC-ROC. After hyperparameter tuning, we evaluated our model on the test set.

In both approaches, we used stratified k-fold cross-validation for train, test, and validation sets, so that each fold had an equal fraction of resistant and susceptible samples. Also, in both approaches, each isolate is used only in one of the training, validation, or testing sets with its resistant/susceptible labels.

3.1.3 Evaluation

In order to evaluate the accuracy of our predictions, we used the AUC-ROC and AUC-PR values. The AUC-ROC metric is the area under the plot of the true positive rate (TPR) against the false positive rate (FPR) at different classification thresholds. The AUC-PR is similar to AUC-ROC, but the plot is that of precision ($\frac{TP}{TP+FP}$) against recall ($\frac{TP}{TP+FN}$) at different classification thresholds.

Since many clinical applications require a predictor with at least a 95% specificity [39], we also evaluate our method using the sensitivity at 95% specificity. This metric tells us what fraction of resistant isolates can be identified as such when that their threshold for prediction is set so as to give at least 95% specificity. Several methods do not achieve a 95% specificity at any threshold, meaning that they cannot be used when a 95% specificity is required.

3.1.4 Hyperparameter optimization

For tuning the model hyperparameters and determining the ideal number of layers for each type of subnetwork (CNN, LSTM, dense layer) in our model we used Bayesian optimization. Bayesian optimization is a method suited to optimizing high-cost functions, such as hyperparameter search for deep learning models, by using a combination of a Gaussian process and Bayesian inference [36]. We used the Python implementation of Bayesian Optimization [27] with 15 iterations to tune the hyperparameters of all the models used in this chapter, with a uniform optimization approach to ensure a fair comparison.

3.1.5 Existing methods

We compared our LRCN model with a large number of existing methods that display state-of-the-art performance. We omitted catalog methods and other methods that have been

found to perform significantly worse than the methods listed here. The following methods were used for the comparison:

- DeepAMR [44]. This method is optimized for predicting drug resistance in TB.
- WnD (Wide-n-Deep neural network) [3]: This method is optimized for predicting drug resistance in TB.
- KOVER (Set Covering Machine algorithm) [10]: A rule-based model to predict drug resistance in multiple bacterial species, including TB.
- GBTs (Gradient-Boosted Trees) [7, 20]: Some publications suggest that this method has the best performance in drug resistance studies, especially in TB.
- LR (Logistic regression) [20]: Some recent research demonstrates their superior performance in predicting drug resistance in TB.
- RF (Random Forests) [19]: Based on some prior research, RFs can achieve very good performance in predicting drug resistance in TB.
- SVM (Support Vector Machines): SVMs are a widely used ML model for binary classification, with applications in many areas.
- LSTM (Long Short-Term Memory) is a widely used neural network model. Note that our LRCN model includes an LSTM component.

Since the publications that propose these models used SNP-based features, we evaluated all the methods separately using SNP-based features and separately using gene burden-based features, to compare the relative performance of these two sets of features with the same class of models in addition to comparing the models to each other (Section 3.2.3).

For all the comparison models we used the exact same data and procedure (e.g. Bayesian Optimization to choose the parameters) that we used for the LRCN. The parameters chosen for the nested cross-validation approach (Section 3.1.2) are listed in Appendix Section C.

Because KOVER produces discrete classifiers and therefore AUC-ROC and AUC-PR of this method can not be achieved at least in a traditional way. In this study, we calculated AUC-ROC using one point, and we were not able to calculate the sensitivity at 95% specificity and AUC-PR. To train KOVER models we used 10-fold cross-validation for hyper-parameter selection. We also set maximum number of rules in the model to 100 as opposed to 10 the default value in KOVER (none of the trained models reached this limitation, since we tested other values as well to get the highest accuracy). Also the possible values for p were: 0.1, 1, 10, 100, 1000 and the chosen values by model were : 10, 1, 1, 10, 1, 1, 10, 1, 1, 10, 1

Moreover, KOVER operates on the presence/absence of k-mers, therefore, we were not able to run it on non-binary gene burden features. Therefore, for training and evaluation of this method, we used SNP features that contain the presence/absence of each SNP in each isolate. As for other steps, we exactly applied them for KOVER as well.

The optimized parameters for the non-nested cross-validation were as follows:

- WnD: 5 layers with 518, 518, 64, 518, and 64 nodes, respectively, and a kernel regularizer value of 0.1.
- GBT: 30 estimators, $min\_samples\_split = 4$, $maxdepth = 130$, and $random\_state = 1$
- LR: the ℓ_2 penalty and $C = 0.1$.
- RF: 140 estimators, a minimum sample split of 4, no bootstrapping, and a maximum depth of 50
- SVM: a linear kernel and $C = 0.1$
- LSTM: 3 LSTM layers, with 355, 455, and 343 nodes, respectively, followed by 4 dense layers with 359, 219, 230, and 147 nodes, respectively.

The parameters chosen by the nested cross-validation approach (Section 3.1.2) are listed in Appendix Section C.

3.1.6 Implementation

We used the Python programming language to implement all the methods in this chapter. We used the Keras library [4] for the deep neural networks, and the Scikit-learn library [30] for the machine learning models.

3.2 Results

Figure 3.3 summarizes the performance of most of our models, its various permutations, and the models we compared it with. In this section we describe each comparison in more detail.

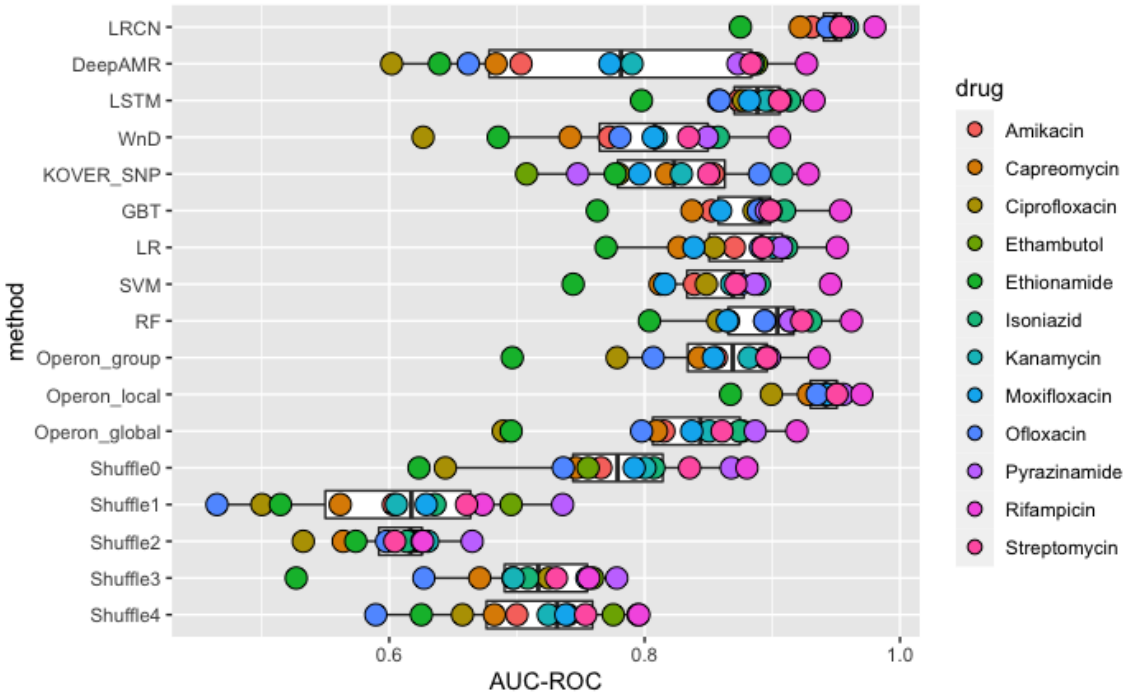
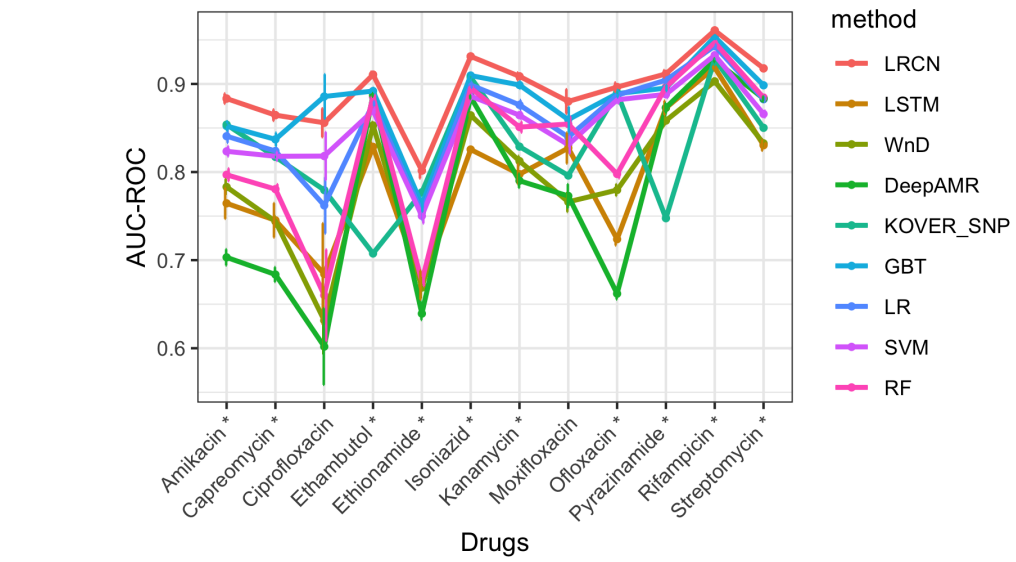
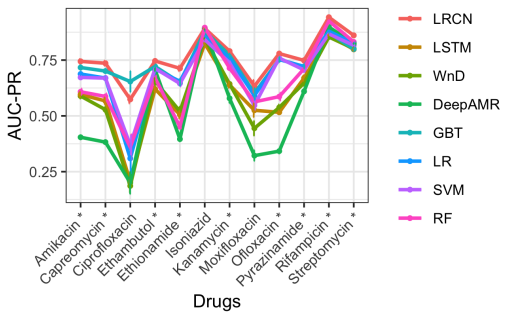


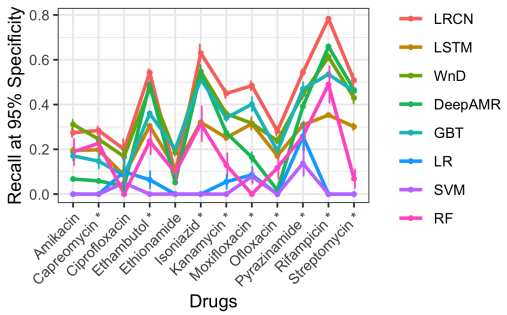
Figure 3.3: Performance of all the methods we tested, non-nested cross-validation approach. The vertical axis lists the methods used, each dot represents a drug, and the horizontal axis shows the AUC-ROC. The white rectangles represent the mean and standard deviation of each method. The "Operon" and "Shuffle" methods are described in Sections 3.2.2 and 3.2.4, respectively.



(a)



(b)



(c)

Figure 3.4: Comparison of LRCN to state-of-the-art models. The error bars represent the confidence intervals on the performance, calculated using a nested cross-validation approach (Section 3.1.2). We calculated 95% confidence interval as the mean ± 1.96 times the standard error across the 10 performance values. Statistically significant differences between LRCN and the second-best method are marked with a "*". See Table 3.1 and Table D.1 for more details of these results. The vertical axis indicates: (a) AUC-ROC, (b) AUC-PR and (c) sensitivity at 95% specificity.

	AUC-ROC	AUC-PR	sensitivity at 95% specificity	AUC-ROC on BCCDC
Streptomycin	0.001	0.001	0.008	0.011
Rifampicin	0.001	0.001	0.001	0.009
Pyrazinamide	0.011	0.001	0.001	0.149
Ofloxacin	0.026	0.002	0.003	-
Moxifloxacin	0.058	0.189	0.001	-
Kanamycin	0.005	0.001	0.001	-
Isoniazid	0.001	0.001 (RF)	0.001	0.002
Ethionamide	0.001	0.001	0.001 (WnD)	-
Ethambutol	0.001	0.001	0.001	0.033
Ciprofloxacin	0.056 (GBT)	0.006 (GBT)	0.290	-
Capreomycin	0.001	0.001	0.005	-
amikacin	0.001	0.001	0.029 (WnD)	-

Table 3.1: The p-values of experiments of Section 3.2.1. The names in parentheses shows the method which performs better than LRCN on that metric.

3.2.1 LRCN outperforms state-of-the-art methods on a large dataset

We evaluated our LRCN method by comparing it to other existing methods, namely, DeepAMR, WnD, KOVER (which uses the SCM algorithm), LSTM, GBT, RF, LR, and SVM (Section 3.1.5). In our evaluation we investigated the following questions:

- Does gene burden-based LRCN outperform other existing methods on the AUC-ROC and AUC-PR metrics? If so, is the difference statistically significant?
- Does a gene burden-based LRCN architecture generalize across multiple data sets?
- How useful is the gene burden-based LRCN for clinical applications? We evaluate this by using a novel metric, the sensitivity at a 95% specificity.

We found that the LRCN method achieves a substantial improvement over all existing methods in most cases. For the AUC-ROC metric, we found that the LRCN method achieves better performance than all the state-of-the-art methods for 10 of the 12 drugs (Figure 3.4a). For the AUC-PR metric, the LRCN method achieves better performance for 9 of the 12 drugs (Figure 3.4b). This performance improvement is statistically significant; a one-sided t-test achieves a p-value below 0.05 (see Table 3.1). The difference in performance between LRCN and other methods on moxifloxacin in both AUC-ROC and AUC-PR, ciprofloxacin in AUC-ROC, and isoniazid in the AUC-PR metric is not statistically significant, so we cannot say whether LRCN performs better or worse than other methods. However, for ciprofloxacin, GBT performs statistically significantly better in AUC-PR. This poor result

on ciprofloxacin may be due to the limited data available for that drug, for which we only have 37 resistant samples.

To gain further confidence in LRCN's robustness and to make sure that our model generalizes to other datasets, we additionally evaluated the trained models on the BCCDC dataset. This data set differs from ReSeqTB and PATRIC in its geographical homogeneity, as it represents a single province in Canada.

We observed that the gene burden-based LRCN performed statistically significantly better on 4 of 5 drugs (Figure 3.5). The difference in performance between LRCN and other methods on pyrazinamide is not statistically significant, which may be because only 134 samples in the BCCDC dataset have a status available for pyrazinamide.

For a prediction of resistance to be usable for clinical diagnosis, it must be made with a high specificity, especially for first-line drugs because the second-line drugs tend to cause more severe side effects and may lead to more frequent treatment failure [47, 11]. Therefore, we also evaluated methods according to the maximum sensitivity they achieve with a minimum of 95% specificity (Section 3.1.3). That is, we find the smallest threshold at which the model has 95% specificity, then we calculate the sensitivity at this threshold. If the model is unable to reach 95% specificity for any threshold, it cannot be used for diagnosis, and therefore scores zero in this metric. The LRCN method has the best sensitivity among all the methods on 9 of the 12 drugs and this difference in performance is statistically significant ($p < 0.05$, one-sided t-test; see Table 3.1). Although for ciprofloxacin, LRCN's performance difference relative to other methods is not statistically significant, and for the two remaining drugs, amikacin and ethionamide, the WnD method achieves a slightly higher sensitivity. Furthermore, many standard ML approaches perform poorly on this metric, which may limit their usefulness in clinical applications (Figure 3.4c).

To evaluate whether the differences in model performance are due to random chance, we calculated the p-values as well as the 95% confidence intervals for each of our three metrics: AUC-ROC, AUC-PR, and sensitivity at 95% specificity. Since we used 10-fold cross-validation, we came up with 10 different model performance metrics for each method (Section 3.1.2). We calculated 95% confidence interval as the mean ± 1.96 times the standard error across the 10 performance values. We calculated the p-values using a one-sided t-test between the vectors of size 10 holding these performance values for our method and the competitor methods.

3.2.2 LRCN exploits gene order

We hypothesized that the improvement in performance in LRCN is due to the importance of gene order. We fed the genes in the same order in which they appear in the M. tuberculosis reference genome, according to MycoBrowser [18]. Indeed, the main difference between the LRCN and the other methods is the combination of CNN and LSTM layers. These layers may help the LRCN to recognize the genes' spatial relationships in the genome.

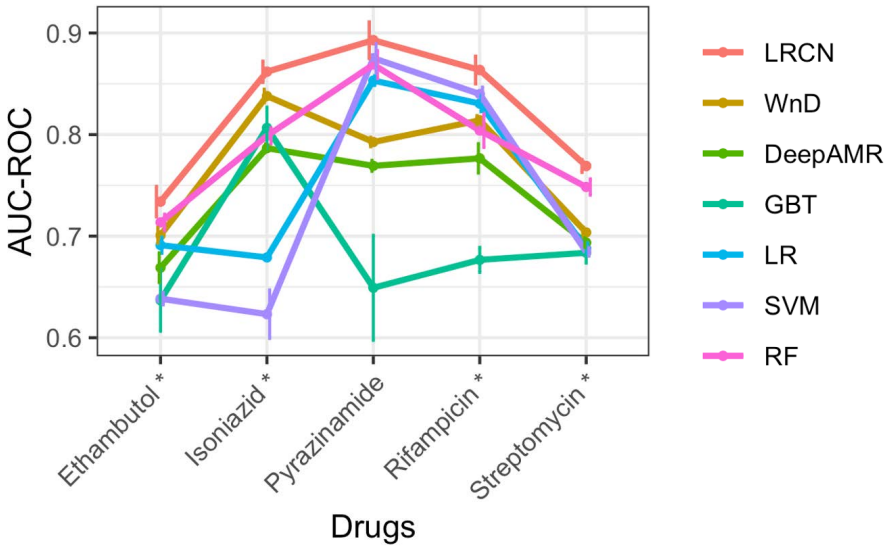


Figure 3.5: Comparison of LRCN to state-of-the-art models on the BCCDC data. Statistically significant differences are marked with a "*".

To test this hypothesis, we randomly permuted the genes to put them in a random order. We then trained and tested our method with the new data. This permutation is analogous to shuffling the pixels of an image. We shuffled the genes' order five times and for each new permutation, we carried out the entire process including Bayesian Optimization hyperparameter tuning.

As expected, we found a significant decrease in the LRCN's performance on the shuffled data, making it comparable to that of a random model. This implies the importance of genes' order in our model (Figure 3.6).

The features' order shouldn't much affect the accuracy of other ML methods such as SVM and LR, since they treat the genome as a "bag of features", without regard for their order.

3.2.3 Gene burden-based features enable a high accuracy on our dataset

To evaluate the efficacy of the gene burden-based approach, we applied the same methods to the SNP-based features (Section 3.1.5). Using SNP-based features, we trained and tested the LRCN and all state-of-the-art methods [44, 3, 10, 7, 20, 19]. We found that the LRCN approach on the SNP-based features performed similarly to other methods (Figure 3.8). However, all SNP-based approaches underperform the gene burden-based LRCN (Figure 3.7). This suggests that using either just gene burden features or just LRCN does not achieve the best performance; both techniques together are required for improved performance.

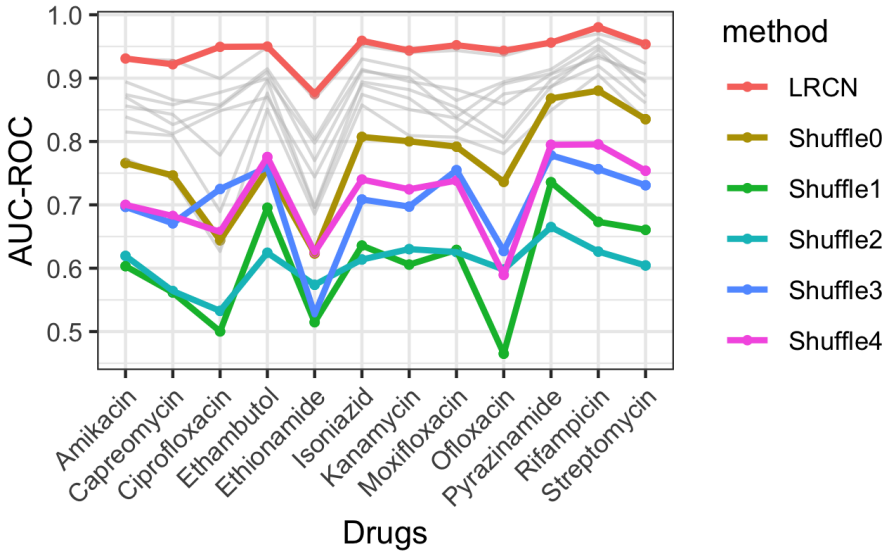


Figure 3.6: Comparison between the test AUC-ROC of our LRCN model using regular and shuffled gene orders. The gray lines represent the result of other models (in Figure **??**, 3.10 as well).

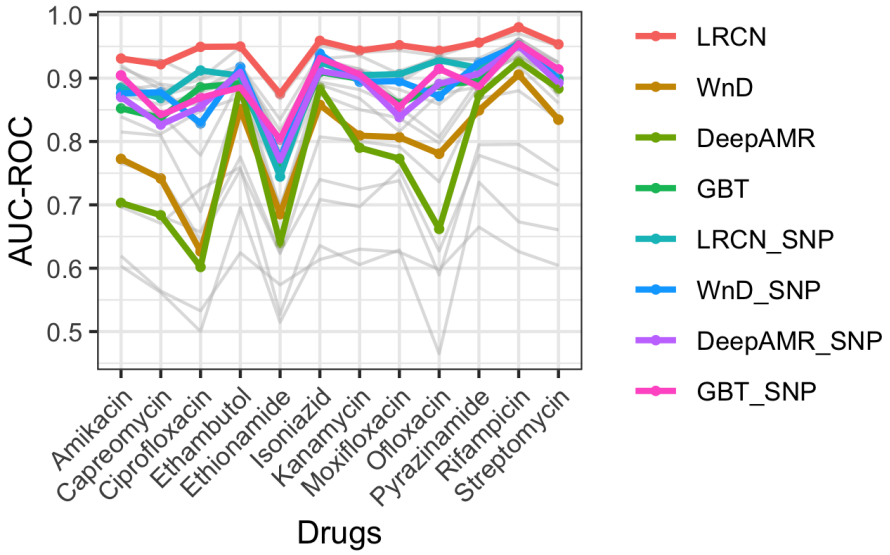


Figure 3.7: Comparison of the gene burden-based and the SNP-based features for LRCN, WnD, DeepAMR, and GBT.

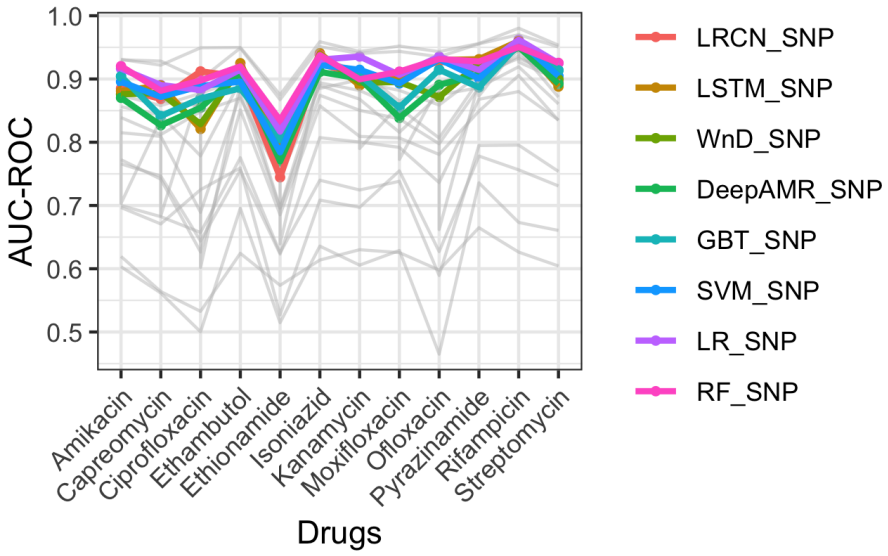


Figure 3.8: Performance with SNP-based features.

3.2.4 The organization of genes into operons drives performance

We hypothesized that the LRCN leverages the fact that neighboring genes in operons have related functions. Operons are clusters of neighbors or co-regulated genes with related functions [29]. We identified operons by assuming that a) all the genes in an operon would also be adjacent in the genome and b) the E. coli-style names of genes belonging to the same operon would share the same initial three letters, but differ in the subsequent letters. Among the 3960 genes with at least one mutation in our data, 879 genes fall within an operon. For this experiment, we shuffled the operon genes in three different ways to disrupt their order (its schematic is shown in Figure 3.9). For each one, we trained and tested our method on the resulting shuffled (permuted) data.

First, we hypothesized that the global order of within-operon genes has a greater effect on the LRCN's results comparing to the order of non-operon genes. To test this, we randomly permuted all the within-operon genes while leaving the other, non-operon, genes intact. We found a significant decrease in the accuracy of the LRCN, similar to that seen when we randomly permute all the genes (Figure 3.10). This suggests the importance of gene order for within-operon genes in comparison to that of non-operon genes.

Second, we assumed that as the operons are typically short (from 2 to 14 genes), the order of genes within each operon should not affect the model's performance, and we should not expect a considerable difference with the original result. To test this, we individually permuted the genes "in place" within each operon, while leaving all the other genes untouched. We observed that, as hypothesized, with this approach the LRCN's performance is comparable to its performance on the original data (Figure 3.10).

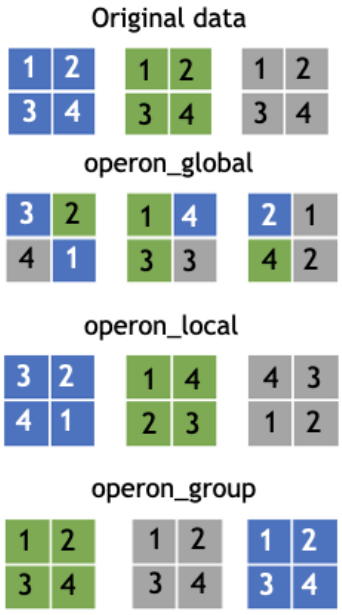


Figure 3.9: A schematic of how we shuffled the data in this section. Let's assume that we have three operons (the big squares) and each operon has 4 genes (the small squares). The "operon_global" is the first approach described in Section 3.2.4, "operon_local" is the second approach, and "operon_group" is the third approach.

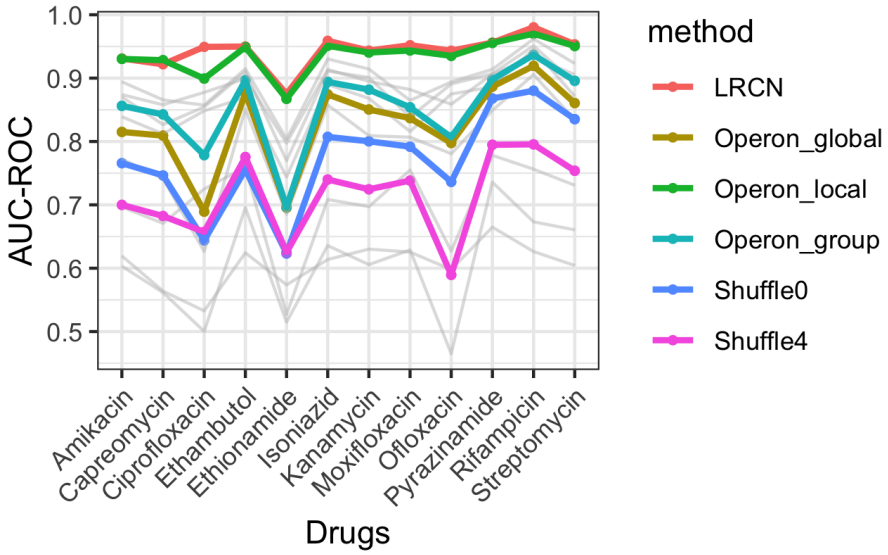


Figure 3.10: Evaluation of LRCN performance on permuted data sets. "operon_global" is the first approach described in Section 3.2.4, "operon_local" is the second approach, and "operon_group" is the third approach. "Shuffle0" and the "Shuffle4" are the highest accuracies from section 3.2.2.

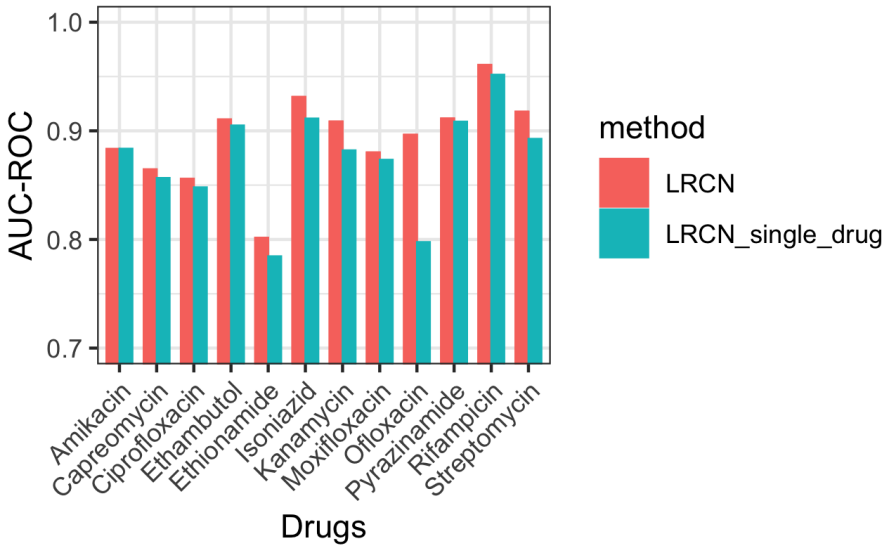


Figure 3.11: Comparison of the performance of the multi-task LRCN and the single-drug LRCN, using gene burden-based features.

Our third hypothesis was that if we keep the gene order fixed within each operon, but permute the locations of the operons, this should negatively affect the accuracy of the LRCN (this is analogous to shuffling the frames of a video, which could render the videos meaningless). We have indeed found a considerable drop in the LRCN's performance in this case (Figure 3.10).

3.2.5 Sharing information between drugs improves performance

We hypothesized in section 2 that using a multi-task model could improve the accuracy of our model, especially for drugs with relatively limited labeled data available, driven by shared mechanisms of resistance. If true, this hypothesis would imply that the patterns learned from one drug can compensate for the lack of training data for another drug. We observed that using the multi-task model improved the performance of the gene burden-based LRCN (Figure 3.11). Furthermore, we see that the gene burden-based LRCN trained on a single drug still displays good performance and can accurately predict drug resistance to that specific drug.

Chapter 4

Interpretable LRCN

The focus of the previous chapter was to achieve high accuracy in predicting drug resistance in TB using LRCN networks. We observed that LRCN could achieve high accuracy and it outperformed the other stat-of-the-art models. However, the lack of interpretability is a considerable disadvantage of the LRCN model as well as most other neural network-based approaches. For this reason, we need to add information to the LRCN method to make it more interpretable and to understand what genes of TB have more effect on the resistance phenotype of an isolate.

4.1 Materials and Methods

4.1.1 Our approach

In order to make the LRCN model interpretable, we used the interpretable model-agnostic explanations (LIME) [33] approach. The way that the LIME approach works is such that it takes each individual sample as well as the model as an input and then it finds the features that have more effect on the decision of the model for that particular sample. Therefore, the output of LIME is a list of features along with their importance score.

Since the LIME approach will output the score of important features for each individual sample, we run it on all the samples that we have, and then the final score of each feature is equal to the average of the scores that this specific feature obtained in each individual sample. Clearly, if a feature is not among the top features of one sample, it will get a score of zero.

4.1.2 Evaluation

In order to be able to evaluate the performance of the model in terms of interpretability, we need to:

- Find what genes of TB have more effect in resistance phenotype from the Biological perspective. Let's call these genes "known DR genes" and the other TB genes the "Uncharacterized genes".

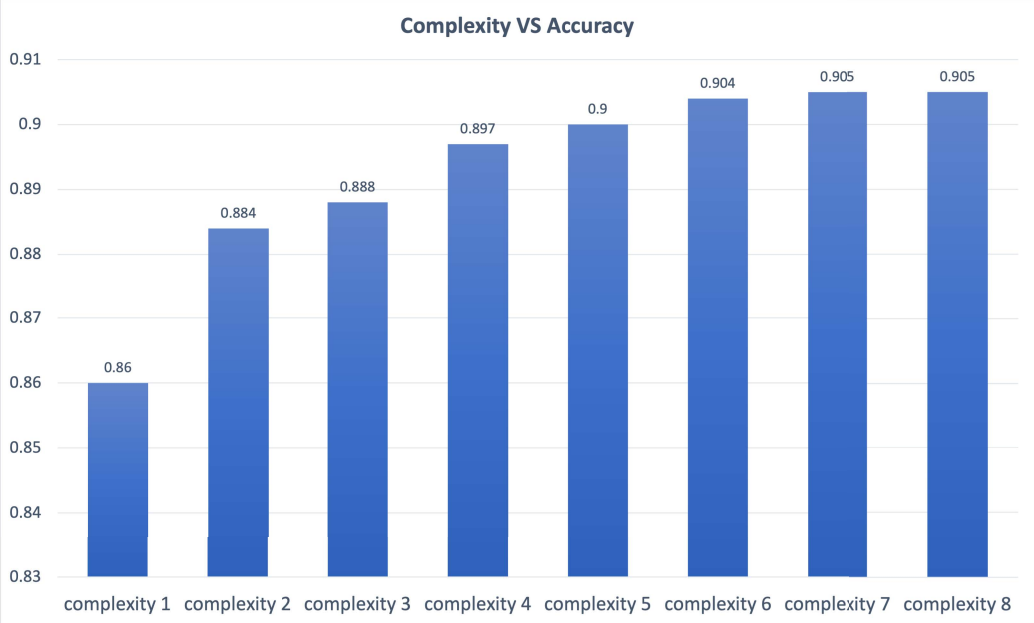


Figure 4.1: Comparison of the performance of the LRCN in different complexities. As it is shown in the figure, at first the performance improved significantly by increasing the complexity of the model, but after some point, the performance does not change significantly.

- Find what genes are more important in predicting drug resistance for the LRCN model.

After having this information then we want to see how many genes that have a high effect on LRCN's output are known DR genes and how many of them are unknown. The LRCN model will have a high performance in terms of interpretability, if it recapitulate the known DR genes.

4.1.3 Implementation

We used the Python programming language to implement all the methods in this chapter. We used the Keras library [4] for the deep neural networks, and the LIME library [33] for the interpretable part models.

4.2 Results

4.2.1 Complexity vs Interpretability

In this experiment, we want to see the relation between the complexity of the LRCN model and its interpretability. We define the LRCN complexity by its number of layers. For instance, if the complexity of the model is one, it means that it has 1layer of each CNN, LSTM, dense layer. Moreover, in this experiment, we want to see the relation between the accuracy of the model and its interpretability results.

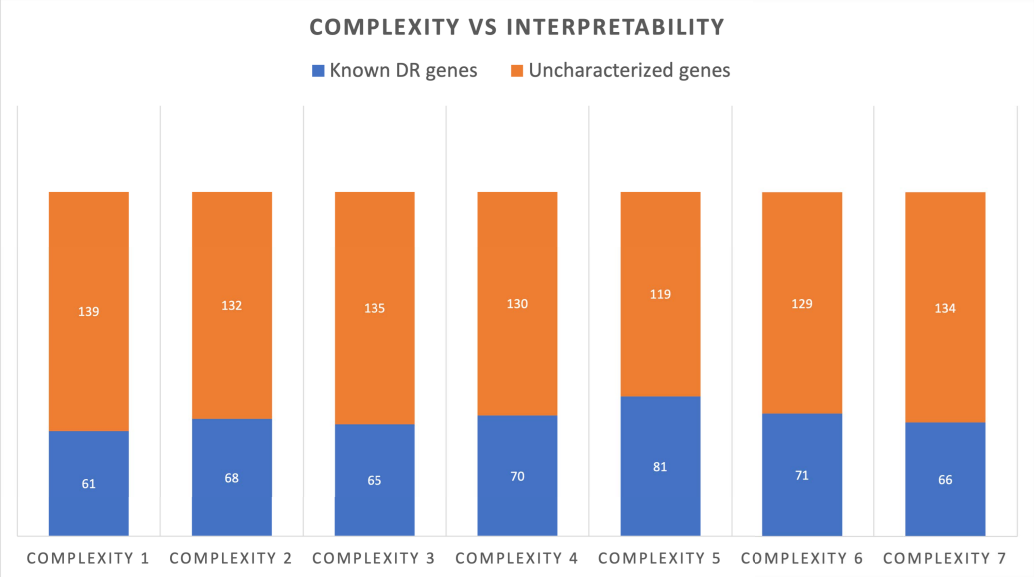


Figure 4.2: Comparison of the interpretability of the LRCN in different complexities. As it is shown in the figure, at first the number of known DR genes increased by increasing the complexity of the model, but after complexity 5, the number of known DR genes drop. This suggests that the LRCN is probably overfitting the data.

In order to see this relation, we trained and tested the LRCN model on 7 different complexity levels from 1 to 7, and then we compare the number of known DR genes that each model considers as important features.

We found that by increasing the complexity of the model, at first both accuracy and the number of known DR genes will increase, but after some point, the accuracy stays quite the same, but the number of known DR genes will decrease (Figures 4.2, 4.1). Although the difference between the number of known DR genes in different complexities is minor, but this suggests that by having a very complex model, the model is probably overfitting on the data and its accuracy is not reliable since it is not using the proper features to do the predictions. However, we can't consider this experiment as a strong support for this hypothesis since we expected that complexity has more effect on the number of known DR genes.

Chapter 5

Conclusion

In this article, we propose the novel use of a deep learning architecture, LRCN, for predicting drug resistance in *M. tuberculosis*. This architecture is a combination of CNN and LSTM layers and thus has the advantage of considering both the gene mutation burden and the gene order.

Our results suggest three major findings:

- Gene burden-based LRCN outperforms the other existing methods in a variety of settings, using three different evaluation metrics, including one relevant for clinical applications. This good performance of the gene burden-based LRCN is robust, as evidenced by a variety of additional experiments including a test on an independent dataset.

- We observed that LRCN achieves its highest performance when we use the gene burden, defined as the number of mutations in each gene, rather than the full SNP data. The gene burden allows us to take into account the importance of individual genes, while the model architecture allows us to consider their order in the genome.

- We found that using the CNN layers before the LSTM layers in our architecture is beneficial for drug resistance prediction, which illustrates the importance of gene locality. We established the importance of gene order with two different experiments. Importantly, these experiments suggest that the good performance of the LRCN model on TB isolates is robust and not explained by chance alone.

In conclusion, we introduced a novel state-of-the-art method for predicting drug resistance by using gene order information alongside gene burden information. Our results suggest that gene burden-based prediction is effective in the context of an LRCN.

This study also suffers from several limitations that we hope to address in future work, namely:

- This article focuses exclusively on *M. tuberculosis* as a proof of concept for the approach. In future work, we plan to apply the LRCN approach to predict the drug resistance in other pathogens such as *Escherichia coli*.

Although in this article, the focus was on achieving the most accurate model for predicting drug resistance. However, the lack of interpretability is a considerable disadvantage of most neural network-based approaches. In order to make the LRCN interpretable, we used the LIME approach and it helped us to see inside the black box of the LRCN model.

The interpretable LRCN suggests two major findings:

- We observed that the LRCN is capable of finding the known DR genes for the resistance phenotype. Although we found that LRCN has lots of Uncharacterized genes among its important features, comparing to the results of other models from other publications [46], the performance of LRCN is acceptable in terms of interpretability and this gives us further confidence in the robustness of LRCN results.

- We found that by increasing the complexity of LRCN, we may get slightly higher accuracy, but we may lose confidence in the LRCN robustness. Since rather than finding the known DR genes, it starts finding Uncharacterized genes which raises the concern of overfitting on the data. However the difference between the number of known DR genes in different complexities is minor and therefore we can't consider this experiment as a strong support for our hypothesis.

Bibliography

[1] P. Bradley, N. Gordon, T Walker, et al. Rapid antibiotic-resistance predictions from genome sequence data for Staphylococcus aureus and Mycobacterium tuberculosis. *Nature Communications*, 6, 2015.

[2] Joshua J Carter, Timothy M Walker, A Sarah Walker, Michael G. Whitfield, Glenn P. Morlock, Timothy EA Peto, James E. Posey, Derrick W Crook, and Philip W Fowler. Prediction of pyrazinamide resistance in mycobacterium tuberculosis using structure-based machine learning approaches. *bioRxiv*, 37, 2019.

[3] Michael L Chen, Akshith Doddi, Jimmy Royer, Luca Freschi, Marco Schito, Matthew Ezewudo, Isaac S Kohane, Andrew Beam, and Maha Farhat. Beyond multidrug resistance: Leveraging rare variants with machine and statistical learning models in mycobacterium tuberculosis resistance prediction. *EBioMedicine*, 43:356–369, 2019.

[4] François Chollet et al. Keras. https://github.com/keras-team/keras, 2015.

[5] F Coll, R McNerney, MD Preston, et al. Rapid determination of anti-tuberculosis drug resistance from whole-genome sequences. *Genome Med.*, 7:51, 2015.

[6] Francesc Coll, Ruth McNerney, José Afonso Guerra-Assunção, Judith R. Glynn, João Perdigão, Miguel Viveiros, Isabel Portugal, Arnab Pain, Nigel Martin, and Taane G. Clark. A robust SNP barcode for typing Mycobacterium tuberculosis complex strains. *Nature Communications*, 5, 2014.

[7] Wouter Deelder, Sofia Christakoudi, Jody Phelan, Ernest Diez Benavente, Susana Campino, Ruth McNerney, Luigi Palla, and Taane Gregory Clark. Machine learning predicts accurately Mycobacterium tuberculosis drug resistance from whole genome sequencing data. *Frontiers in Genetics*, 10:922, 2019.

[8] Christopher A Desjardins, Keira A Cohen, Vanisha Munsamy, Thomas Abeel, Kashmeel Maharaj, Bruce J Walker, Terrance P Shea, Deepak V Almeida, Abigail L Manson, Alex Salazar, et al. Genomic and functional analyses of mycobacterium tuberculosis strains implicate ald in d-cycloserine resistance. *Nature genetics*, 48(5):544–551, 2016.

[9] Jeff Donahue, Lisa Anne Hendricks, Marcus Rohrbach, Subhashini Venugopalan, Sergio Guadarrama, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39:667–691, 2017.

[10] Alexandre Drouin, Gaël Letarte, Frédéric Raymond, Mario Marchand, Jacques Corbeil, and François Laviolette. Interpretable genotype-to-phenotype classifiers with performance guarantees. *Scientific Reports*, 9(1):4071, Mar 2019.

[11] D. Falzon, E. Jaramillo, H. J. Schunemann, M. Arentz, M. Bauer, J. Bayona, L. Blanc, J. A. Caminero, C. L. Daley, C. Duncombe, C. Fitzpatrick, A. Gebhard, H. Getahun, M. Henkens, T. H. Holtz, J. Keravec, S. Keshavjee, A. J. Khan, R. Kulier, V. Leimane, C. Lienhardt, C. Lu, A. Mariandyshev, G. B. Migliori, F. Mirzayev, C. D. Mitnick, P. Nunn, G. Nwagboniwe, O. Oxlade, D. Palmero, P. Pavlinac, M. I. Quelapio, M. C. Raviglione, M. L. Rich, S. Royce, S. Rusch-Gerdes, A. Salakaia, R. Sarin, D. Sculier, F. Varaine, M. Vitoria, J. L. Walson, F. Wares, K. Weyer, R. A. White, and M. Zignol. WHO guidelines for the programmatic management of drug-resistant tuberculosis: 2011 update. *European Respiratory Journal*, 38(3):516–528, August 2011.

[12] Maha R Farhat, Luca Freschi, Roger Calderon, Thomas Ioerger, Matthew Snyder, Conor J Meehan, Bouke de Jong, Leen Rigouts, Alex Sloutsky, Devinder Kaur, et al. GWAS for quantitative resistance phenotypes in Mycobacterium tuberculosis reveals resistance genes and regulatory regions. *Nature communications*, 10(1):1–11, 2019.

[13] Silke Feuerriegel, Viola Schleusener, Patrick Beckert, Thomas A. Kohl, Paolo Miotto, Daniela M. Cirillo, Andrea M. Cabibbe, Stefan Niemann, and Kurt Fellenberg. PhyResSE: a web tool delineating Mycobacterium tuberculosis antibiotic resistance and lineage from whole-genome sequencing data. *Journal of Clinical Microbiology*, 53(6):1908–1914, 2015.

[14] Guo Liang Gan, Matthew Nguyen, Elijah Willie, Brian Lee, Cedric Chauve, Maxwell Libbrecht, and Leonid Chindelevitch. Geographic heterogeneity impacts drug resistance predictions in Mycobacterium tuberculosis. *bioRxiv*, 27, 2020.

[15] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November 1997.

[16] H. Iwai, M. Kato-Miyazawa, T Kirikae, and T. Miyoshi-Akiyama. CASTB (the comprehensive analysis server for the Mycobacterium tuberculosis complex): A publicly accessible web server for epidemiological analyses, drug-resistance prediction and phylogenetic comparison of clinical isolates. *Tuberculosis*, 1:843–844, 2015.

[17] Suha Kadura, Nicholas King, Maria Nakhoul, Hongya Zhu, Grant Theron, Claudio U Köser, and Maha Farhat. Systematic review of mutations associated with resistance to the new and repurposed Mycobacterium tuberculosis drugs bedaquiline, clofazimine, linezolid, delamanid and pretomanid. *Journal of Antimicrobial Chemotherapy*, 75, 05 2020. dkaa136.

[18] Adamandia Kapopoulou, Jocelyne M Lew, and Stewart T Cole. The MycoBrowser portal: a comprehensive and manually annotated resource for mycobacterial genomes. *Tuberculosis*, 91(1):8–13, 2011.

[19] Samaneh Kouchaki, Yang Yang, Alexander Lachapelle, Timothy M. Walker, A. Sarah Walker, CRyPTIC Consortium , Timothy E. A. Peto, Derrick W. Crook, and David A. Clifton. Multi-label random forest model for tuberculosis drug resistance classification and mutation ranking. *Frontiers in Microbiology*, 11:667, 2020.

[20] Samaneh Kouchaki, Yang Yang, Timothy M Walker, A Sarah Walker, Daniel J Wilson, Timothy E A Peto, Derrick W Crook, CRyPTIC Consortium, and David A Clifton.

Application of machine learning techniques to tuberculosis drug resistance analysis. *Bioinformatics*, 35(13):2276–2282, 11 2018.

[21] Raski Leinonen, Hideaki Sugawara, and Martin Shumway. The Sequence Read Archive. *Nucleic Acids Res.*, 39:D19–21, 2011.

[22] Rasko Leinonen, Ruth Akhtar, Ewan Birney, Lawrence Bower, Ana Cerdeno-Tárraga, et al. The European Nucleotide Archive. *Nucleic Acids Research*, 39:D28–31, 2011.

[23] H Li. Aligning sequence reads, clone sequences and assembly contigs with BWA-MEM. *arXiv*, 3, 2013.

[24] H Li, Handsaker B, Wysoker A, Fennell T, Ruan J, Homer N, Marth G, Abecasis G, Durbin R, and 1000 Genome Project Data Processing Subgroup. The Sequence Alignment/Map format and SAMtools. *Bioinformatics*, 25(16):2078–2079, 2009.

[25] Niall McLaughlin, Jesus Martinez del Rincon, and Paul Miller. Recurrent convolutional network for video-based person re-identification. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1325–1334, USA, June 2016. IEEE.

[26] Tra-My Ngo and Yik-Ying Teo. Genomic prediction of tuberculosis drug-resistance: benchmarking existing databases and prediction algorithms. *BMC bioinformatics*, 20(1):68, 2019.

[27] Fernando Nogueira. Bayesian Optimization: Open source constrained global optimization tool for Python, 2014–.

[28] Jim O'Neill. Antimicrobial resistance: Tackling a crisis for the health and wealth of nations. Technical report, Review on Antimicrobial Resistance, 2014.

[29] Anne E Osbourn and Ben Field. Operons. *Cellular and Molecular Life Sciences*, 66(23):3755–3775, 2009.

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

[31] Ryan Poplin, Valentin Ruano-Rubio, Mark A. DePristo, Tim J. Fennell, Mauricio O. Carneiro, Geraldine A. Van der Auwera, David E. Kling, Laura D. Gauthier, Ami Levy-Moonshine, David Roazen, Khalid Shakir, Joel Thibault, Sheila Chandran, Chris Whelan, Monkol Lek, Stacey Gabriel, Mark J Daly, Ben Neale, Daniel G. MacArthur, and Eric Banks. Scaling accurate genetic variant discovery to tens of thousands of samples. *bioRxiv*, 22, 2017.

[32] Mario C Raviglione and Ian M Smith. XDR tuberculosis—implications for global public health. *New England Journal of Medicine*, 356(7):656–659, 2007.

[33] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016*, pages 1135–1144, 2016.

[34] V. Schleusener, C. Köser, P. Beckert, et al. Mycobacterium tuberculosis resistance prediction and lineage classification from genome sequencing: comparison of automated analysis tools. *Scientific Reports*, 7, 2017.

[35] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *CoRR*, abs/1404.7828, 2014.

[36] Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. Practical bayesian optimization of machine learning algorithms. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, page 2951–2959, Red Hook, NY, USA, 2012. Curran Associates Inc.

[37] Angela M Starks, Enrique Avilés, Daniela M Cirillo, Claudia M Denkinger, David L Dolinger, Claudia Emerson, Jim Gallarda, Debra Hanna, Peter S Kim, Richard Liwski, et al. Collaborative effort for a centralized worldwide tuberculosis relational sequencing data platform. *Clinical Infectious Diseases*, 61(suppl_3):S141–S146, 2015.

[38] A Steiner, D Stucki, M Coscolla, S Borrell, and S Gagneux. KvarQ: targeted and direct variant calling from fastq reads of bacterial genomes. *BMC Genomics*, 15, 2014.

[39] Michelle Su, Sarah W. Satola, and Timothy D. Read. Genome-based prediction of bacterial antibiotic resistance. *Journal of clinical microbiology*, 57(3):e01405–18, Feb 2019.

[40] Andrej Trauner, Sonia Borrell, Klaus Reither, and Sebastien Gagneux. Evolution of drug resistance in tuberculosis: Recent progress and implications for diagnosis and therapy. *Drugs*, 10, 2014.

[41] Alice R Wattam, David Abraham, Oral Dalay, Terry L Disz, Timothy Driscoll, Joseph L Gabbard, Joseph J Gillespie, Roger Gough, Deborah Hix, Ronald Kenyon, et al. PATRIC, the bacterial bioinformatics database and analysis resource. *Nucleic acids research*, 42(D1):D581–D591, 2014.

[42] WHO. Antimicrobial resistance: global report on surveillance. Technical report, World Health Organization, 2014.

[43] WHO. Global tuberculosis report 2019. Technical report, World Health Organization, 2019.

[44] Yang Yang, Timothy M Walker, A Sarah Walker, Daniel J Wilson, Timothy E A Peto, Derrick W Crook, Farah Shamout, CRyPTIC Consortium, Tingting Zhu, and David A Clifton. DeepAMR for predicting co-occurrent resistance of Mycobacterium tuberculosis. *Bioinformatics*, 35(18):3240–3249, 01 2019.

[45] Han Yuan, Ivan Paskov, Hristo Paskov, Alvaro J. González, and Christina S. Leslie. Multitask learning improves prediction of cancer drug sensitivity. *Scientific Reports*, 6(1):31619, Aug 2016.

[46] Hooman Zabeti, Nick Dexter, Amir Hosein Safari, Nafiseh Sedaghat, Maxwell Libbrecht, and Leonid Chindelevitch. An interpretable classification method for predicting drug resistance in M. tuberculosis. *bioRxiv*, 18, 2020.

[47] Lilia E Ziganshina, Albina F Titarenko, and Geraint R Davies. Fluoroquinolones for treating tuberculosis (presumed drug-sensitive). *Cochrane Database of Systematic Reviews*, 72, June 2013.

Appendix A

Related works

Title, Author, Short description

1. Predicting antimicrobial resistance using conserved genes, Marcus Nguyen, AMR via conserved genes
2. Genomic prediction of tuberculosis drugresistance: benchmarking existing databases and prediction algorithms, Tra-My Ngo, Benchmarking TB databases
3. DeepAMR for predicting co-occurrent resistance of Mycobacterium tuberculosis, Yang Yang, DeepAMR paper
4. Genetic Determinants of Drug Resistance in Mycobacterium tuberculosis and Their Diagnostic Value, Maha R. Farhat, DecisionTreesForTB.
5. Deciphering drug resistance in Mycobacterium tuberculosis using wholegenome sequencing: progress, promise, and challenges, Keira A. Cohen, Deciphering drug resistance in Mycobacterium tuberculosis
6. Prediction of Susceptibility to First-Line Tuberculosis Drugs by DNA Sequencing, , CRyPTiC susceptibility predictions
7. Combining Structure and Genomics to Understand Antimicrobial Resistance, Tanushree Tunstall, Combining Structure and Genomics to Understand Antimicrobial Resistance
8. Beyond multidrug resistance: Leveraging rare variants with machine and statistical learning models in Mycobacterium tuberculosis resistance prediction, Michael L. Chen,
9. Prediction of antibiotic resistance in Escherichia coli from large-scale pan-genome data, Danesh Moradigaravand, E coli resistance prediction
10. Prediction of Acquired Antimicrobial Resistance for Multiple Bacterial Species Using Neural Networks, Aytan-Aktug, Drug resistance via DNN

11. Evaluation of Whole-Genome Sequence Method to Diagnose Resistance of 13 Anti-tuberculosis Drugs and Characterize Resistance Genes in Clinical Multi-Drug Resistance Mycobacterium tuberculosis Isolates From China, Xinchang Chen, Drug resistance prediction tools

12. Deciphering drug resistance in Mycobacterium tuberculosis using wholegenome sequencing: progress, promise, and challenges, Keira A. Cohen, DR-TB via WGS

13. Prediction of rifampicin resistance beyond the RRDR using structure-based machine learning approaches, Stephanie Portelli, DR via structure-based machine learning

14. Rapid inference of antibiotic resistance and susceptibility by genomic neighbour typing, Karel Břinda , DR by genomic neighbour typing

15. Detection of low-frequency resistance-mediating SNPs in next-generation sequencing data of Mycobacterium tuberculosis complex strains with binoSN, Viola Dreyer, Detection of low-level TB resistance

16. Interpreting k-mer–based signatures for antibiotic resistance prediction, Magali Jaillard, K-mer interpretation

17. Interpretable Model for Antibiotic Resistance Prediction In Bacteria Using Deep Learning, MANOJ JHA, Interpretable Model for Antibiotic Resistance Prediction In Bacteria Using Deep Learning

18. Improved Resistance Prediction in Mycobacterium tuberculosis by Better Handling of Insertions and Deletions, Premature Stop Codons, and Filtering of Non-informative Sites, Camilla Hundahl Johnsen, Improved Resistance Prediction in Mycobacterium tuberculosis

19. GWAS for quantitative resistance phenotypes in Mycobacterium tuberculosis reveals resistance genes and regulatory regions, Maha R. Farhat, GWAS for quantitative resistance phenotypes in Mycobacterium tuberculosis reveals resistance genes

20. Genetics and roadblocks of drug resistant tuberculosis, João Perdigão, Genetics-and-roadblocks-of-drug-resistant-tu-2019-Infection–Genetics-and-Ev

21. The role of whole genome sequencing in antimicrobial susceptibility testing of bacteria: report from the EUCAST Subcommittee, M.J. Ellington, EUCAST report

22. Rapid genomic first- and second-line drug resistance prediction from clinical Mycobacterium tuberculosis specimens using Deeplex®-MycTB, Silke Feuerriegel, ERJ paper on rapid DR prediction

23. pncA gene mutations in reporting pyrazinamide resistance among the MDRTB suspects, Xiaoyuan Wu, pncA-gene-mutations-in-reporting-pyrazinamide-resi-2019-Infection–Genetics

24. Next-Generation Sequencing Approaches to Predicting Antimicrobial Susceptibility Testing Results, Rebecca Yee, NGS for DR prediction

25. WGS to predict antibiotic MICs for Neisseria gonorrhoeae, David W. Eyre, Neisseria dataset with MICs

26. Using machine learning to predict antimicrobial minimum inhibitory concentrations and associated genomic features for nontyphoidal Salmonella, Marcus Nguyen, MIC prediction in Salmonella

27. Multi-Label Random Forest Model for Tuberculosis Drug Resistance Classification and Mutation Ranking, Samaneh Kouchaki ,

28. A large scale evaluation of TBProfiler and Mykrobe for antibiotic resistance prediction in Mycobacterium tuberculosis, Pierre Mahé, Mahe Tournoud comparison of TB tools

29. Application of machine learning techniques to tuberculosis drug resistance analysis, Samaneh Kouchaki, LogisticRegressonForTB

30. Large scale modeling of antimicrobial resistance with interpretable classifiers, Alexandre Drouin, Kover latest paper

31. Interpretable genotype-tophenotype classifers with performance guarantees, Alexandre Drouin, Kover paper

32. Developing an in silico minimum inhibitory concentration panel test for Klebsiella pneumoniae, Marcus Nguyen, Klebsiella MICs

33. Robust detection of point mutations involved in multidrug-resistant Mycobacterium tuberculosis in the presence of co-occurrent resistance markers, Julian Libiseller-Egger, Robust detection of point mutations involved in MDR-TB

34. Systematic review of mutations associated with resistance to the new and repurposed Mycobacterium tuberculosis drugs bedaquiline, clofazimine, linezolid, delamanid and pretomanid, Suha Kadura, Resistance to new TB drugs

35. Machine learning with random subspace ensembles identifies antimicrobial resistance determinants from pan-genomes of three pathogens, Jason C. HyunI, Random subspace ensembles for AMR

36. Recent advances in molecular diagnostics and understanding mechanisms of drug resistance in nontuberculous mycobacterial diseases, Hee Jae Huh, Recent-advances-in-molecular-diagnostics-and-understandi-2019-Infection–Gen

37. Prediction of pyrazinamide resistance in Mycobacterium tuberculosis using structure-based machine learning approaches., Joshua J Carter, Pyrazinamide resistance in TB

38. Predicting antimicrobial resistance in Pseudomonas aeruginosa with machine learning-enabled molecular diagnostics, Ariane Khaledi, Pseudomonas genomics + transcriptomics of DR

39. Predicting Phenotypic Polymyxin Resistance in Klebsiella pneumoniae through Machine Learning Analysis of Genomic Data, Nenad Macesic, Predicting resistance in Klebsiella

40. Whole-Genome Sequencing for Drug Resistance Profile Prediction in Mycobacterium tuberculosis, Sebastian M. Gygli, WGS for MTB DR

41. Evaluation of parameters affecting performance and reliability of machine learning-based antibiotic susceptibility testing from whole genome sequencing data, Allison L. Hicks,

42. Structure guided prediction of Pyrazinamide resistance mutations in pncA, Malancha Karmakar, Structure-guided PZA resistance prediction

43. Survey of drug resistance associated gene mutations in Mycobacterium tuberculosis, ESKAPE and other bacterial species,AbhirupaGhosh,SurveyOfMutations

44. Machine Learning Predicts Accurately Mycobacterium tuberculosis Drug Resistance From Whole Genome Sequencing Data, Wouter Deelder, Taane Clark's latest paper on ML for DR

45. Prediction of Phenotypic Antimicrobial Resistance Profiles From Whole Genome Sequences of Non-typhoidal Salmonella enterica, Saskia Neuert,

46. Whole-genome sequencing for prediction of Mycobacterium tuberculosis drug susceptibility and resistance: a retrospective cohort study, Timothy M Walker, Tim Walker's heuristic paper

47. VAMPr: VAriant Mapping and Prediction of antibiotic resistance via explainable features and machine learning, Jiwoong Kim, VAMPr paper

48. Use of whole genome sequencing for detection of antimicrobial resistance: Mycobacterium tuberculosis, a model organism., Vincent Escuyer, WGS for AMR - TB as a case study

Appendix B

Data

	Streptomycin	*Rifampicin*	*Pyrazinamide*	*Ofloxacin*	*Moxifloxacin*	*Kanamycin*	*Isoniazid*	*Ethionamide*	*Ethambutol*	*Ciprofloxacin*	*Capreomycin*	*amikacin*
Streptomycin	2104	1860	601	597	91	552	1989	342	1158	16	462	449
Rifampicin	1860	2968	667	750	117	654	2840	432	1342	35	522	542
Pyrazinamide	601	667	754	255	81	247	684	196	492	24	277	242
Ofloxacin	597	750	255	800	122	487	747	303	457	25	331	333
Moxifloxacin	91	117	81	122	129	85	113	67	51	18	66	74
Kanamycin	552	654	247	487	85	697	641	318	472	21	402	440
Isoniazid	1989	2840	684	747	113	641	3445	448	1378	31	512	530
Ethionamide	342	432	196	303	67	318	448	498	323	13	235	226
Ethambutol	1158	1342	492	457	51	472	1378	323	1407	22	402	399
Ciprofloxacin	16	35	24	25	18	21	31	13	22	37	12	8
Capreomycin	462	522	277	331	66	402	512	235	402	12	552	431
amikacin	449	542	242	333	74	440	530	226	399	8	431	573

Table B.1: The pairwise similarity of the resistant status for each pair of drugs. Each cell represents the number of same samples between resistant isolates of that two drugs.

	Streptomycin	*Rifampicin*	*Pyrazinamide*	*Ofloxacin*	*Moxifloxacin*	*Kanamycin*	*Isoniazid*	*Ethionamide*	*Ethambutol*	*Ciprofloxacin*	*Capreomycin*	*amikacin*
Streptomycin	3021	2410	1525	1101	643	1104	2225	500	2651	85	657	726
Rifampicin	2410	4747	2651	893	506	844	4141	387	3608	212	545	614
Pyrazinamide	1525	2651	3104	708	401	509	2343	182	2820	214	673	594
Ofloxacin	1101	893	708	2111	731	1350	821	649	1321	48	1198	1237
Moxifloxacin	643	506	401	731	832	639	490	233	629	35	455	689
Kanamycin	1104	844	509	1350	639	1739	778	574	1206	187	915	970
Isoniazid	2225	4141	2343	821	490	778	4289	348	3208	59	512	554
Ethionamide	500	387	182	649	233	574	348	1018	605	26	454	481
Ethambutol	2651	3608	2820	1321	629	1206	3208	605	4689	354	872	893
Ciprofloxacin	85	212	214	48	35	187	59	26	354	406	89	60
Capreomycin	657	545	673	1198	455	915	512	454	872	89	1439	1113
amikacin	726	614	594	1237	689	970	554	481	893	60	1113	1460

Table B.2: The pairwise similarity of the susceptible status for each pair of drugs. Each cell represents the number of same samples between susceptible isolates of that two drugs.

Appendix C

Model parameters

Layer \ Model	LRCN1	LRCN2	LRCN3	LRCN4	LRCN5	LRCN6	LRCN7	LRCN8	LRCN9	LRCN10
CNN1 (kernel, filter, pool)	(4, 5, 4)	(6, 8, 6)	(3, 8, 3)	(3, 8, 3)	(6, 8, 4)	(6, 8, 6)	(3, 8, 3)	(3, 8, 3)	(6, 8, 4)	(5, 7, 5)
CNN2 (kernel, filter, pool)	(5, 4, 4)	(3, 8, 3)	(6, 8, 6)	(4, 4, 4)	(3, 7, 3)	(6, 4, 4)	(6, 8, 6)	(6, 8, 4)	(6, 8, 6)	(4, 5, 4)
CNN3 (kernel, filter, pool)	(5, 7, 4)	-	(3, 7, 3)	-	-	(3, 4, 3)	(3, 4, 3)	(6, 4, 4)	(3, 8, 3)	(5, 7, 4)
CNN4 (kernel, filter, pool)	(4, 6, 4)	-	(3, 4, 3)	-	-	-	(3, 8, 3)	-	(3, 4, 3)	(6, 4, 4)
CNN5 (kernel, filter, pool)	-	-	(6, 8, 4)	-	-	-	(6, 4, 4)	-	(6, 8, 4)	-
LSTM1	478	64	64	439	478	457	64	413	518	518
LSTM2	466	518	71	86	451	227	518	497	64	446
LSTM3	350	518	518	518	460	350	454	518	-	-
LSTM4	444	303	64	309	518	-	-	456	-	-
LSTM5	-	518	64	518	518	-	-	518	-	-
Dense1	306	518	518	159	356	518	64	515	518	277
Dense2	293	64	64	518	416	518	95	362	467	76
Dense3	325	64	-	518	286	-	-	518	-	348
Dense4	-	64	-	64	-	-	-	77	-	-
Dense5	-	64	-	317	-	-	-	415	-	-

Table C.1: Parameters of LRCN models. Each row represents a layer and each column represents a model. The layers order in the table is the same of their order in the model.

Layer \ Model	LSTM1	LSTM2	LSTM3	LSTM4	LSTM5	LSTM6	LSTM7	LSTM8	LSTM9	LSTM10
layer LSTM1	518	361	311	355	353	353	265	478	404	478
layer LSTM2	64	142	401	453	237	237	116	466	482	237
layer LSTM3	-	247	365	343	281	281	135	350	-	291
layer LSTM4	-	291	-	-	-	-	313	444	-	444
layer LSTM5	-	-	-	-	-	-	-	-	-	365
Dense1	518	467	228	359	484	484	252	306	69	306
Dense2	64	470	347	219	480	480	395	293	70	480
Dense3	64	132	154	230	239	239	442	325	171	-
Dense4	518	488	404	147	-	-	390	-	453	-
Dense5	64	-	88	244	-	-	-	-	305	-

Table C.2: Parameters of LSTM models. Each row represents a layer and each column represents a model. The layers order in the table is the same of their order in the model.

Layer \ Model	WnD1	WnD2	WnD3	WnD4	WnD5	WnD6	WnD7	WnD8	WnD9	WnD10
Dense1	64	64	64	64	518	64	198	362	64	64
Dense2	518	132	244	518	64	306	518	64	518	444
Dense3	64	64	518	-	-	293	518	286	-	481
Dense4	518	518	64	-	-	-	518	171	-	518
Dense5	488	488	64	-	-	-	518	518	-	317
kernel regularizer	0.1	0.1	0.1	0.01	0.1	0.01	0.1	0.01	0.1	0.01

Table C.3: Parameters of WnD models. Each row represents a layer and each column represents a model. The layers order in the table is the same of their order in the model.

	RF1	RF2	RF3	RF4	RF5	RF6	RF7	RF8	RF9	RF10
n_estimators	110	130	140	60	130	140	130	110	120	110
min_samples_split	4	4	4	4	4	3	4	4	4	3
bootstrap	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE	FALSE
max_depth	50	50	None	50	70	50	30	50	None	80

Table C.4: Parameters of RF models. Each row represents a layer and each column represents a model.

	LR1	LR2	LR3	LR4	LR5	LR6	LR7	LR8	LR9	LR10
C	1	0.1	1	1	0.1	1	1	1	1	1
max_iter	657.7915	1000	657.7915	79.43	955.89	657.791473	657.791473	1000	1000	657.791473
penalty	l2	l1	l1	l1	l2	l1	l1	l2	l1	l1
solver	newton-cg	liblinear	liblinear	liblinear	sag	liblinear	liblinear	newton-cg	liblinear	liblinear

Table C.5: Parameters of LR models. Each row represents a layer and each column represents a model.
The possible values for penalty were: l1 (solver = liblinear), l2 (solver = newton-cg, lbfgs, sag), elasticnet (solver = saga), none. The possible values for C were: (0.01, 0.1, 1, 10, 100)

	SVM1	SVM2	SVM3	SVM4	SVM5	SVM6	SVM7	SVM8	SVM9	SVM10
C	0.1	1	0.1	0.1	0.1	1	0.1	0.1	0.01	0.1
kernel	linear	linear	linear	linear	linear	linear	linear	linear	linear	linear
gamma	-	-	-	-	-	-	-	-	-	-
degree	-	-	-	-	-	-	-	-	-	-

Table C.6: Parameters of SVM models. Each row represents a layer and each column represents a model.
The possible values for kernel were: linear, poly, rbf. The possible values for C were: (0.01, 0.1, 1, 10, 100)

	GBT1	GBT2	GBT3	GBT4	GBT5	GBT6	GBT7	GBT8	GBT9	GBT10
max_depth	130	140	90	60	70	90	50	60	50	70
min_samples_split	4	4	3	2	2	3	2	2	4	2
n_estimators	30	30	20	20	50	20	90	20	10	50
random_state	1	1	-	0	-	-	1	0	-	-

Table C.7: Parameters of GBT models. Each row represents a layer and each column represents a model.

Appendix D

LRCN results

		Streptomycin	Rifampicin	Pyrazinamide	Ofloxacin	Moxifloxacin	Kanamycin	Isoniazid	Ethionamide	Ethambutol	Ciprofloxacin	Capreomycin	amikacin
LRCN	AUC-ROC	0.918 ± 0.003	0.961 ± 0.002	0.911 ± 0.006	0.896 ± 0.008	0.880 ± 0.022	0.909 ± 0.006	0.931 ± 0.002	0.802 ± 0.014	0.911 ± 0.005	0.856 ± 0.026	0.865 ± 0.010	0.883 ± 0.009
LRCN	AUC-PR	0.861 ± 0.004	0.942 ± 0.003	0.749 ± 0.015	0.779 ± 0.012	0.633 ± 0.045	0.790 ± 0.009	0.892 ± 0.004	0.713 ± 0.021	0.746 ± 0.010	0.574 ± 0.029	0.736 ± 0.018	0.744 ± 0.017
LRCN	R-95%-S	0.508 ± 0.050	0.783 ± 0.014	0.544 ± 0.021	0.282 ± 0.033	0.484 ± 0.033	0.450 ± 0.033	0.629 ± 0.063	0.095 ± 0.007	0.543 ± 0.025	0.2 ± 0.076	0.284 ± 0.029	0.275 ± 0.032
WnD	AUC-ROC	0.832 ± 0.007	0.902 ± 0.005	0.858 ± 0.007	0.779 ± 0.010	0.766 ± 0.018	0.812 ± 0.009	0.864 ± 0.003	0.668 ± 0.019	0.852 ± 0.006	0.631 ± 0.062	0.743 ± 0.011	0.783 ± 0.014
WnD	AUC-PR	0.801 ± 0.008	0.853 ± 0.009	0.647 ± 0.020	0.537 ± 0.023	0.444 ± 0.054	0.643 ± 0.021	0.845 ± 0.007	0.520 ± 0.034	0.655 ± 0.024	0.186 ± 0.049	0.527 ± 0.021	0.588 ± 0.020
WnD	R-95%-S	0.430 ± 0.043	0.613 ± 0.025	0.445 ± 0.032	0.238 ± 0.023	0.316 ± 0.046	0.359 ± 0.028	0.548 ± 0.019	0.186 ± 0.024	0.473 ± 0.024	0.168 ± 0.057	0.245 ± 0.025	0.311 ± 0.036
DeepAMR	AUC-ROC	0.883 ± 0.004	0.927 ± 0.003	0.873 ± 0.008	0.662 ± 0.011	0.773 ± 0.022	0.790 ± 0.013	0.885 ± 0.003	0.639 ± 0.012	0.888 ± 0.003	0.602 ± 0.070	0.684 ± 0.014	0.703 ± 0.015
DeepAMR	AUC-PR	0.824 ± 0.005	0.898 ± 0.004	0.610 ± 0.022	0.342 ± 0.018	0.322 ± 0.040	0.579 ± 0.036	0.887 ± 0.003	0.396 ± 0.016	0.707 ± 0.012	0.200 ± 0.082	0.382 ± 0.015	0.404 ± 0.015
DeepAMR	R-95%-S	0.465 ± 0.017	0.659 ± 0.016	0.391 ± 0.031	0.017 ± 0.005	0.165 ± 0.038	0.274 ± 0.031	0.541 ± 0.059	0.052 ± 0.014	0.489 ± 0.023	0.035 ± 0.035	0.059 ± 0.014	0.068 ± 0.014
LSTM	AUC-ROC	0.830 ± 0.010	0.918 ± 0.003	0.872 ± 0.013	0.723 ± 0.011	0.827 ± 0.028	0.797 ± 0.018	0.825 ± 0.004	0.671 ± 0.026	0.828 ± 0.007	0.684 ± 0.092	0.745 ± 0.031	0.764 ± 0.028
LSTM	AUC-PR	0.797 ± 0.010	0.861 ± 0.006	0.669 ± 0.022	0.516 ± 0.014	0.525 ± 0.051	0.638 ± 0.025	0.824 ± 0.008	0.505 ± 0.029	0.620 ± 0.022	0.205 ± 0.016	0.567 ± 0.026	0.595 ± 0.022
LSTM	R-95%-S	0.301 ± 0.027	0.353 ± 0.007	0.308 ± 0.016	0.171 ± 0.017	0.314 ± 0.037	0.251 ± 0.021	0.320 ± 0.006	0.114 ± 0.017	0.304 ± 0.014	0.083 ± 0.083	0.197 ± 0.019	0.194 ± 0.022
GBT	AUC-ROC	0.898 ± 0.004	0.953 ± 0.003	0.895 ± 0.006	0.889 ± 0.007	0.860 ± 0.023	0.899 ± 0.007	0.909 ± 0.003	0.763 ± 0.013	0.892 ± 0.006	0.886 ± 0.040	0.837 ± 0.012	0.852 ± 0.009
GBT	AUC-PR	0.800 ± 0.005	0.872 ± 0.002	0.721 ± 0.015	0.753 ± 0.019	0.601 ± 0.040	0.771 ± 0.013	0.881 ± 0.004	0.650 ± 0.026	0.722 ± 0.015	0.654 ± 0.076	0.701 ± 0.018	0.717 ± 0.020
GBT	R-95%-S	0.460 ± 0.011	0.535 ± 0.006	0.469 ± 0.052	0.192 ± 0.019	0.401 ± 0.053	0.340 ± 0.023	0.513 ± 0.006	0.201 ± 0.026	0.360 ± 0.021	0.083 ± 0.112	0.147 ± 0.055	0.171 ± 0.036
KOVER	AUC-ROC	0.850 ± 0.000	0.928 ± 0.000	0.748 ± 0.000	0.890 ± 0.000	0.796 ± 0.000	0.829 ± 0.000	0.908 ± 0.000	0.777 ± 0.000	0.708 ± 0.000	0.780 ± 0.000	0.817 ± 0.000	0.854 ± 0.000
KOVER	AUC-PR	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
KOVER	R-95%-S	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
RF	AUC-ROC	0.884 ± 0.004	0.947 ± 0.002	0.896 ± 0.008	0.797 ± 0.008	0.854 ± 0.013	0.851 ± 0.010	0.895 ± 0.004	0.675 ± 0.008	0.885 ± 0.003	0.660 ± 0.083	0.780 ± 0.009	0.796 ± 0.011
RF	AUC-PR	0.831 ± 0.006	0.925 ± 0.003	0.711 ± 0.015	0.586 ± 0.029	0.562 ± 0.036	0.714 ± 0.023	0.895 ± 0.003	0.451 ± 0.016	0.670 ± 0.019	0.359 ± 0.095	0.587 ± 0.023	0.608 ± 0.024
RF	R-95%-S	0.069 ± 0.069	0.489 ± 0.133	0.258 ± 0.106	0.117 ± 0.078	0.0 ± 0.0	0.129 ± 0.086	0.313 ± 0.128	0.095 ± 0.064	0.236 ± 0.096	0.0 ± 0.0	0.227 ± 0.094	0.188 ± 0.097
LR	AUC-ROC	0.882 ± 0.004	0.943 ± 0.003	0.904 ± 0.004	0.887 ± 0.004	0.839 ± 0.012	0.876 ± 0.008	0.898 ± 0.005	0.754 ± 0.016	0.877 ± 0.004	0.762 ± 0.051	0.823 ± 0.007	0.840 ± 0.012
LR	AUC-PR	0.825 ± 0.006	0.879 ± 0.003	0.720 ± 0.009	0.752 ± 0.011	0.590 ± 0.040	0.759 ± 0.010	0.857 ± 0.005	0.654 ± 0.022	0.709 ± 0.009	0.309 ± 0.112	0.669 ± 0.013	0.687 ± 0.017
LR	R-95%-S	0.0 ± 0.0	0.0 ± 0.0	0.257 ± 0.105	0.0 ± 0.0	0.086 ± 0.058	0.055 ± 0.055	0.0 ± 0.0	0.0 ± 0.0	0.064 ± 0.064	0.1 ± 0.1	0.0 ± 0.0	0.0 ± 0.0
SVM	AUC-ROC	0.865 ± 0.006	0.932 ± 0.002	0.888 ± 0.006	0.881 ± 0.006	0.830 ± 0.016	0.864 ± 0.006	0.886 ± 0.003	0.750 ± 0.013	0.869 ± 0.005	0.818 ± 0.043	0.817 ± 0.008	0.823 ± 0.009
SVM	AUC-PR	0.815 ± 0.007	0.867 ± 0.003	0.705 ± 0.013	0.759 ± 0.012	0.547 ± 0.033	0.738 ± 0.010	0.841 ± 0.004	0.644 ± 0.023	0.712 ± 0.010	0.369 ± 0.071	0.669 ± 0.015	0.672 ± 0.010
SVM	R-95%-S	0.0 ± 0.0	0.0 ± 0.0	0.137 ± 0.091	0.0 ± 0.0	$0.078 + -0.053$	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.05 ± 0.05	0.0 ± 0.0	0.0 ± 0.0

Table D.1: The performance and confidence intervals of all models in Figure 3.

Appendix E

How to run the codes

This is a brief explanation of how to run the codes which we used in this article. Please feel free to open an issue on our GitHub repository if you have any other questions.

E.1 LRCN-drug-resistance

In the “LRCN-drug-resistance” repository, you can find the methods that are related to the ML models of this research and the main method is Long-term Recurrent Convolutional Network (LRCN).

1. pipe_line_gene.py: This is the main class for running the models on the gene-based dataset, it simply loads the data using "data_preprocess.process()" and then run the proper model on it.

2. pipe_line.py: This is the main class for running the models on the SNP-based dataset, it simply loads the data using "data_preprocess.process()" and then run the proper model on it.

3. create_dataset/gene_dataset_creator.py: This code creates the gene dataset, using the SNP-based dataset.

4. create_dataset/operons_finder.py This code finds the operon indexes in the gene-based dataset.

5. create_dataset/shuffle_operon_dataset_creator.py: We shuffled the features in this code as we explained in the paper.

6. loading_data package: Codes here are for loading the proper dataset. You need to call the "data_preprocess.process()" function for this purpose. parameters are as follow:

   num_of_files: this is for the SNP-based data, you can specify the number of files that you want to load. nrow: num of rows which are loaded. limited: default is False, if you set it True, five of the drugs would be drop (ciprofloxacin, capreomycin, amikacin,

ethionamide, moxifloxacin) gene_dataset: default is False, if you set it True then the gene-based data would be loaded.

You don't need to modify or call the other methods.

7. models/model_gene_based.py: you need to run the "run_bayesian()" function with proper data as input. it will run the proper functions in other classes and will print the output.

8. models/Bayesian_optimizer.py: This file has the main implementation of the model. Which is the implementation of LRCN with K-fold and the Bayesian optimization.

E.2 Genotype-collector-and-SNP-dataset-creator

In the "Genotype-collector-and-SNP-dataset-creator" repository, you can find the methods that are related to the Data section of this research.

In this code, we do the following things:

1. We download the .fastq files from the european nucleotide archive (ENA) for each isolate.

2. We find the SNPs for each isolate by using the bwa-mem and then samtools and GATK.

3. We create a binary table. Each row represents an isolate and each column represents the SNP.

Here is a brief explanation of the methods of this repository.

1. dataCollector.py In this code by using the isolate id in the "FILE_NAME".txt file we will download its .fastq files from the ENA database.

2. SNP.py In this code by using the isolate id in the "FILE_NAME".txt file and its .fastq files we will find its SNPs using bwa-mem and samtools and GATK. Then we get the common SNPs of both approaches (samtools and GATK) and store them in ("Final_" + id + ".vcf").

   If the code fails to find SNPs of special isolate, the id will be written in the "missIdForSNP.txt"

   By calling preprocessing() function you will get the ("Final_" + id + "_table.csv") file which contatin "pos, ref, alt" for each SNP of that isolate.

3. tableCreatorGit.py In this code, we create a binary table. Each row represents an isolate and each column represents the SNP.

4. sparseMatrix.py Will store the table of tableCreatorGit.py in the sparse format to reduce size.

E.3 M.tuberculosis-dataset-for-drug-resistant

In the “M.tuberculosis-dataset-for-drug-resistant” repository, you can find the datasets that we used in this research.