DeepMDRP: A Multi-task CNN based approach towards resistance prediction in MTB isolates

Kushagra Pandey, Avesh Agrawal, Randeepkumar Sahu
Indian Institute of Technology, Kanpur
`kushagrap20@iitk.ac.in, aavesh@iitk.ac.in, randeeps@iitk.ac.in`

Abstract

In this report we explore the problem of predicting phenotypic susceptibility of MTB isolates towards 10 drugs including 4 first line drugs and 4 second line drugs. We experiment with a number of statistical models for predicting MTB resistance towards different drugs and propose a Deep Convolutioal Neural Network that achieves state of the art results in predicting MTB resistance in all the first and second line drugs while achieving competitive performance on other drugs. We also give a detailed report of several approaches that were critical in the performance of our method while also giving a detailed account of approaches that did not perform as well, hoping that it would be helpful to the research community. The source code for our approach is publicly available at *https://github.com/kpandey008/drug-resistance-prediction*

1 Introduction

Tuberculosis is one of the top 10 leading causes of mortality worldwide and the leading cause of death from a single infectious agent, caused by the bacillus *Mycobacterium Tuberculosis (MTB)* [7]. To make the situation even more difficult to tackle, there has been a significant rise of drug-resistant bacterial strains due to the increasing use of antibacterial drugs. Moroever, the phenotypic drug susceptibility test (DST) that determines which drugs are effective towards particular TB infections is slow and costly.

Recently, methods involving Whole Genome Sequencing(WGS) of MTB have emerged as a faster alternative for identifying drug-resistant TB [11]. A number of Single Nucleotide Polymorphisms (SNP) have been identified that are associated with drug resistance found in MTB isolates. However, the interactions between different SNP's associated with drug resistance in MTB isolates is complex and cannot be modeled efficiently using rule-based methods that incorporate significant domain knowledge when predicting MTB resistance towards a particular drug. More recently, there has been a surge in using Machine learning(ML) methods for tackling this problem as a predictive modeling problem where the SNP's for MTB isolates are used as features and the model needs to predict if the isolate is resistant to a particular drug. [17] provide a comparison between several classical ML methods like Logistic Regression, Random Forests (RF) and Support Vector Machines (SVM) to predict MTB resistance to 11 assayed drugs including 4 first line drugs (Rifampicin (RIF), Isoniazid (INH), Ethambutol (EMB), Pyrazinamide (PZA)) and 6 second line drugs (Streptomycin (STR), Capreomycin (CAP), Ciprofloxacin (CIP), Kanamycin (KAN), Moxifloxacin(MOXI), Amikacin (AMK), Ofloxacin (OFX)). [1] pose the resistance prediction problem as a multi-task learning problem where they use a Deep neural network (WDNN) to predict resistance of MTB isolates for all the 11 drugs using a single neural network. [18] pretrain a Deep de-noising autoencoder to learn latent representations over the binary SNP features and use the resulting autoencoder in composition with a classifier to predict MTB resistance towards the first-line drugs.

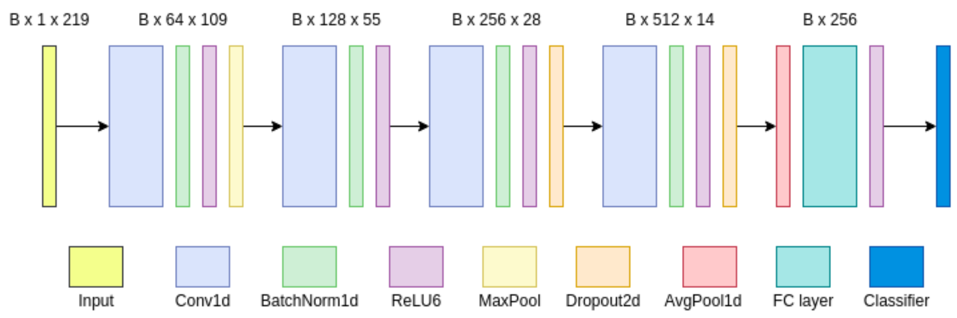

Figure 1: Network Architecture. The dimensions above each block represent the shape of the output of that module . The output of the Conv module is of the form (B x C x d) where B is the batch size, C is the number of channels, and d is the feature length. The output of the FC module is (B x d).

In this report, we propose DeepMDRP, a multi-task Deep Convolutional Neural Network (CNN) for predicting resistance of MTB isolates towards 10 drugs (excluding CIP). We provide several justifications for our model choice and show that our approach outperforms existing approaches when predicting MTB resistance in most first and second line drugs. The rest of the report is organized as follows:

2 Deep MTB Drug Resistance Predictor (DeepMDRP)

In this section we present an overview of our network. As shown in Figure 1 our network takes a batch of feature vectors as an input of the form $B \times 1 \times d$ where B is the batch size and d is the dimension of the feature vector. Here, each feature vector is representative of a single MTB isolate and each dimension of the feature vector is a binary value which represents whether a mutation occurs in that isolate (1) or not (0). This feature vector is passed through a Deep CNN which outputs a prediction vector of size equal to the number of drugs considered in the analysis. Therefore, the output prediction vector would be a 11 dimensional vector that outputs the probabililty estimates of resistance of the MTB isolate of each drug in the order: $\{RIF, INH, PZA, EMB, STR, CIP, CAP, AMK, MOXI, OFLX, KAN\}$

2.1 Network Architecture

The network architecture for the Deep CNN is shown in Figure 1. It takes a $B \times d$ dimensional input where B is the batch size and d is a 219 dimensional feature vector. The CNN network consists of four 1-d convolutional blocks followed by a fully connected block and a classifier layer. The input B x 219 dimensional feature vector is first reshaped into a B x 1 x 219 dimensional tensor which is then input to the network. As shown in Figure 1, we use BatchNorm1d [3] and Dropout2d [14] modules to regularize training. We use the ReLU6 activation as opposed to ReLU as it gave better empirical performance during evaluation. We choose CNN's as a prediction model for this problem in contrast to other ML models for the following reasons:

- CNN's can operate on sequences of varying length as they are independent of the input size. This allows us to experiment with transfer learning approaches which have been successfully applied to several problems in vision and NLP. Note that this is not possible with a simple deep MLP which is dependent of the size of the input features.
- CNN's promote parameter sharing which has two advantages. It allows us to detect recurring patterns in sequences and allows for a reduction in the parameter count. This further enables us to build deeper models that can capture complex patterns between mutations.
- CNN's build more abstract features at each layer and thus are able to learn more complex patterns. We hypothesize that this behavior might be important to model complex co-occuring patterns between different mutations.

2.2 Training and Testing

The training procedure for our network is described in Algorithm 1. One important aspect of our training algorithm is the use of Mixup [19] data augmentation. As we discuss in the experimental details, Mixup leads to an improved overall performance of the network and prevents the network from overfitting. We choose the masked binary cross entropy loss as the loss criterion as the dataset contains -1 labels for some drugs which needs to be masked during training. We use the Adam [4] optimizer for weight update and the gradient updates are made through backpropagating through the network. The training Hyperparameters are discussed in section 3.2
The testing procedure is simple as it involves only a single forward pass of the test sample through the trained network.

Algorithm 1: Algorithm for training DeepMDRP

```
Input  : Training set $D$, Initialized model: $M$, Hyperparameters: $\{num\_epochs, \alpha, \eta, \beta_1, \beta_2\}$
Output : Trained model $M$

for $i \leftarrow 1$ to $num\_epochs$ do
    for $X_{batch}, Y_{batch} \in D$ do
        $mixup \sim \mathcal{U}(0,1)$;
        if $mixup > 0.5$ then
            $\lambda \sim Beta(\alpha, \alpha)$;
            $\hat{X} = permute(X_{batch})$;
            $\hat{Y} = permute(Y_{batch})$;
            $X_{mixup} = \lambda X_{batch} + (1-\lambda)\hat{X}$;
            $Y_{pred} = \sigma(M(X_{mixup}))$;
            $loss = \lambda * binary\_cross\_entropy(Y_{pred}, Y_{batch}) + (1-\lambda) *$
              $binary\_cross\_entropy(Y_{pred}, \hat{Y})$
        else
            $Y_{pred} = \sigma(M(X_{batch}))$;
            $loss = binary\_cross\_entropy(Y_{pred}, Y_{batch})$
        end
        $w \leftarrow Adam(\eta, \beta_1.\beta_2)$
    end
end
```

3 Experimental Details

3.1 Dataset

The dataset used for training consists of 3393 samples with 222 features. In the provided dataset some features have missing values. More specifically, the features $SNP\_CN\_2714366\_C967A\_V323L\_eis$, $SNP\_I\_2713795\_C329T\_inter\_Rv2415c\_eis$, $SNP\_I\_2713872\_C252A\_inter\_Rv2415c\_eis$ have missing values. We remove these features from the dataset resulting in 219 binary features (Each feature representing a mutation). The targets provided in the dataset also have missing values which we mask during training. We use a validation set of 792 samples taken from [1] as the authors in this paper use the same feature set as ours to evaluate their model performance.

3.2 Implementation Details

Model: The model architecture as shown in Figure 1 is a multi-task architecture that consists of 4 Conv1d layers containing 64, 128, 256, 512 filters respectively. Each conv layer is followed by a BatchNorm1d layer and a ReLU6 activation. The first conv layer also uses a MaxPool1d layer after applying the ReLU6 activation. In the last two Conv blocks we use a Dropout2d module with a drop probability of 0.3. The output of the last conv block is fed into a AveragePooling layer followed by a flatten operation. The resulting output is then fed into a fully-connected layer -> ReLU6 -> BN unit. The final output is then fed into a classifier which makes predictions for individual drugs.

Table 1: Model performance comparison on the Multitask prediction task.

Models	Avg. AUC
DeepCNN	94.5
WDNN-256	94.5
WDNN-512	**94.7**
DeepCNN (WS + GN)	94.4
DeepCNN-SE	94.4
DeepCNN-MH (8 heads)	93.7
DeepCNN-NMF	81.6
DeepCNN-kPCA	66.8

Training: We use PyTorch [8] for developing the neural net framework developmnent and training. We train our DeepCNN predictor for 100 epochs with a batch size of 32 using the Adam optimizer. The mixup parameter α is set to 2.0. The initial learning rate is set to 0.001 and weight decay to 0.0001. We adjust the learning rate using the Cosine Annealing scheduler with warm restarts [5] ($T_0 = 10$ and $t_{mul} = 2$). For computing loss, we use a custom masked Binary cross entropy (BCE) loss without any weight balancing. We train the entire system on the Google Colab platform.

3.3 Ablation Studies

For the results in this section, we report the AUC scores on our validation set of 792 isolates. We evaluate the model performance on the best performing checkpoint for each method and do not perform any ensembling for any of the methods. We do not change the training settings for these experiments unless mentioned otherwise.

3.3.1 Choice of Models

Apart from using a Deep-CNN as a model for our experiments we also tried a number of other approaches as follows:

- We created a deep-MLP containing three fc blocks (Linear -> BatchNorm1d -> ReLU Dropout(0.5)) This architecture is similar to the WDNN used in [1] so we denote this network architecture as WDNN. We experiment with two variants of WDNN. One with all hidden layers of dimension 256 and one with 512. We refer to these models as *WDNN-256* and *WDNN-512* respectively.
- We also experimented with self-attention based models [15]. The primary intuition behind using these models was to model correlations between mutations as these models have recently been shown to be extremely efficient in modeling long term dependencies in sequences in several NLP and computer vision tasks. We implement a MultiHead attention module and use this layer at the end of the conv blocks in the DeepCNN. We denote this model as *DeepCNN-MH*.
- We also experimented with different layer normalization methods. More specifically we tried using a Weight standardization [10] + GroupNorm [16] (WS + GN) in combination with the DeepCNN. The intuition behind this choice is based on the fact that WS + GN has been shown to stabilize gradients during training which might improve performance. We denote this model by DeepCNN (WS + GN)
- Another variant that we tried is based on using Squeeze and Excitation [2] (SE) units in combination with the DeepCNN architecture. We denote this model by DeepCNN-SE.
- Lastly, we tried involved pre-processing the input features using dimensionality reduction methods and then using the pre-processed features as inputs to our model. We specifically tried two such methods: kernel-PCA and NMF and denote these methods as *DeepCNN-kPCA* and *DeepCNN-NMF* respectively. The dimension of the features using these methods was reduced to 100.

The performance of this model on multi-task drug resistance prediction is shown in Table 1 As can be observed, any kind of pre-processing on the data severely affects the performance of the model.

Table 2: Model performance comparison for different loss functions.

Masked Loss	Avg. AUC
BCE	**94.5**
MSE	93.7
MAE	93.7
OHEM-BCE	93.8

Table 3: Effect of weight balancing methods

Method	Avg. AUC
Unweighted	**94.5**
Inverse	93.8
Pos-Neg	93.9
Median	93.7

Out of all the models, WDNN-512 performs the best with a average AUC score of 94.7. DeepCNN and WDNN-256 perform similiarly on the prediction task. It was surprising to see DeepCNN with multi-head attention not performing as expected. We hypothesize that this architecture needs more experimentation. Also MultiHead attention performs really well on a large amount of data, which is not the case here. This could be another reason for the architecture achieving lesser AUC scores.

3.3.2 Choice of Loss function

Apart from using the Masked Binary Cross Entropy Loss (BCE), we also experimented with Masked versions of Mean Squared Loss (MSE), L1 Loss (MAE), and Online Hard Example Mining (OHEM)[12] version of BCE. OHEM selects samples from the training batch for which the computed loss is the highest and backpropagates through only these hard examples. We used a value of 15 hard examples for each drug. The model used for this experiment was the Multitask Deep-CNN predictor. The results for these variants are given in Table 2. As evident from the results, masked BCE performs the best among all the variants so we use it for our final model.

3.3.3 Dataset imbalance

The training data used for our experiments suffers from the common problem of class imbalance. We hypothesized that some form of weight balancing would be needed during training. Due to this reason, we experimented with three types of weight balancing schemes during training.

- *Inverse Frequency Balancing* : This computes the normalized inverse class frequencies for each drug.
- *Median frequency balancing* which computes the inter-class median frequency and computes class weights by $median/class\_freq$ for each drug. In this way minority classes have weights greater than 1 and majority classes have weights < 1.
- Lastly, we also experimented by measuring the ratio of the frequencies of the positive and the negative samples and using those weights for the positive class during training for each drug. This is equivalent to subsampling methods in which we can either oversample the minority class or undersample the majority class.

However, none of these schemes worked better than the unweighted version of our classifier. The results are shown in Table 3.

3.3.4 Choice of Optimizer

Apart from the Adam optimizer, we experimented with Stochastic Gradient Descent (SGD), RMSProp and AdamW [6] optimizers. The learning rate for all the optimizers was set to 0.001. We found similiar performance for different kinds of optimizers. However, SGD took longer to converge so we selected Adam as the final choice for the model.

3.3.5 Choice of Learning Rate Scheduler

We use the Cosine Annealing learning rate scheduler with warm restarts [5]. We also experimented with a Poly learning rate policy and a Step scheduler.

Table 4: Finetuning vs Train-from-scratch

Drug	With Fine-tuning	W/o fine-tuning
RIF	98.6	**98.7**
INH	**97.4**	97.1
PZA	**90.6**	89.3
EMB	**92.6**	91.7

Table 5: Effect of Mixup

Model	With Mixup	W/o Mixup
DeepCNN	94.5	93.9
WDNN256	94.5	94.3
WDNN512	94.7	94.2

Table 6: WDNN[] vs DeepCNN performance

Drug	WDNN	DeepCNN
RIF	98.2	**98.4**
INH	95.9	**97.2**
PZA	88.3	**89.0**
EMB	**92.2**	91.6
STR	**94.2**	93.8
CAP	**80.8**	77.5
AMK	95.0	**97.7**
MOXI	90.2	**95.6**
OFLX	86.6	**86.9**
KAN	87.9	**89.6**

Table 7: Kaggle Leaderboard scores

Drug	Avg AUC
RIF	98.95
INH	97.99
PZA	92.50
EMB	94.47
STR	96.18
CAP	87.55
AMK	98.05
MOXI	98.08
OFLX	89.94
KAN	93.21

3.3.6 Regularization strategies

Since, our training dataset size is relatively small (only 3393) samples, we experiment with a number of regularization strategies. Apart from Mixup and Dropout2d, we tried Dropout [13], Label Smoothing [9] and adding noise to the input data. However, Mixup plays a very important role in improving the performance of the models as demonstrated in Table 5. It is worth noting that the DeepCNN model benefits the most from Mixup.

3.3.7 Impact of transfer learning

As pointed in section 2, using a Deep-CNN can enable strategies like Transfer learning which can be very efficient when using less amount of labeled data. Transfer learning involves fine-tuning a classifier which has been pre-trained on a similiar source task so that a target task with less data can benefit from the training of the source task trained with a large amount of data. For pre-training, we train our Deep-CNN classifier on the DeepAMR [18] dataset which consists of 8390 samples with 5824 mutations for predicing MTB resistance to the four first line drugs. We then fine-tune our classifier on our target dataset to predict MTB resistance towards the assayed drugs. The results are shown in Table 4 As evident from the results, transfer learning certainly improves the performance of INH, EMB and PZA as compared to its train-from-scratch DeepCNN counterpart.

4 Final Performance

In this section we compare the results of our multitask DeepCNN based approach with WDNN as specified in [1] as this is most relevant to our work. We use the same validation set which is used by [1] in their evaluation. The results are shown in Table 6. As evident from the results, our multi-task method beats the state of the art method performance on the majority of the drugs. We also submitted our evaluations to the Kaggle leaderboard as a part of the ongoing competition. The results are shown in Table 7. It is worth noting that the results submitted to the Kaggle competitions were submitted using multiple variants of WDNN256, WDNN512 and DeepCNN.

5 Conclusion and Future Work

In this report we present a novel DeepCNN based method with mixup augmentation for predicting drug resistance in MTB isolates. We show that our method achieves better performance compared to the state of the art model on multiple first and second line drugs. We also present an overview of different model variants that we experimented with and a detailed ablation study covering several aspects of model development including choice of loss functions, optimizers, learning rate schedulers and regularization strategies . To conclude, we see the following directions as direct extensions to our work.

- More experimentation with multi-head attention based models for modeling correlations between mutations.
- Since our dataset contains a lot of unlabeled data for different drugs. It would be interesting to explore the application of semi-supervised learning methods in the multi-label context.
- Another direction could be to explore model / checkpoint averaging methods as a way to make the model predictions more robust which we have not explored as a part of this work.

6 Contributions

Kushagra contributed to:

- Conducting literature review in MTB drug resistance and semi-supervised learning methods.
- Developing and testing the core network architecture presented in this report
- Ablation Studies (choice of loss functions, model variants etc.)
- Report Development

References

[1] 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. ISSN 2352-3964. doi: https://doi.org/10.1016/j.ebiom.2019.04.016. URL `http://www.sciencedirect.com/science/article/pii/S2352396419302506`.

[2] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu. Squeeze-and-excitation networks, 2019.

[3] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. pages 448–456, 2015. URL `http://jmlr.org/proceedings/papers/v37/ioffe15.pdf`.

[4] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[5] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts, 2017.

[6] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019.

[7] World Health Organization. *Global tuberculosis report 2019*. World Health Organization, 2019.

[8] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library, 2019.

[9] Gabriel Pereyra, George Tucker, Jan Chorowski, Łukasz Kaiser, and Geoffrey Hinton. Regularizing neural networks by penalizing confident output distributions, 2017.

[10] Siyuan Qiao, Huiyu Wang, Chenxi Liu, Wei Shen, and Alan Yuille. Micro-batch training with batch-channel normalization and weight standardization, 2020.

[11] T. Quan, Zharain Bawa, Dona Foster, Tim Walker, Carlos del Ojo, Priti Rathod, Zamin Iqbal, Phelim Bradley, Janet Mowbray, A. Walker, Derrick Crook, David Wyllie, Timothy Peto, and E. Smith. Evaluation of whole-genome sequencing for mycobacterial species identification and drug susceptibility testing in a clinical setting: a large-scale prospective assessment of performance against line probe assays and phenotyping. *Journal of Clinical Microbiology*, 56: JCM.01480–17, 11 2017. doi: 10.1128/JCM.01480-17.

[12] Abhinav Shrivastava, Abhinav Gupta, and Ross Girshick. Training region-based object detectors with online hard example mining, 2016.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL `http://jmlr.org/papers/v15/srivastava14a.html`.

[14] Jonathan Tompson, Ross Goroshin, Arjun Jain, Yann LeCun, and Christoph Bregler. Efficient object localization using convolutional networks. *CoRR*, abs/1411.4280, 2014. URL `http://arxiv.org/abs/1411.4280`.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[16] Yuxin Wu and Kaiming He. Group normalization, 2018.

[17] Yang Yang, Katherine E Niehaus, Timothy M Walker, Zamin Iqbal, A Sarah Walker, Daniel J Wilson, Tim E A Peto, Derrick W Crook, E Grace Smith, Tingting Zhu, and David A Clifton. Machine learning for classifying tuberculosis drug-resistance from DNA sequencing data. *Bioinformatics*, 34(10):1666–1671, 12 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx801. URL `https://doi.org/10.1093/bioinformatics/btx801`.

[18] 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. ISSN 1367-4803. doi: 10.1093/bioinformatics/btz067. URL `https://doi.org/10.1093/bioinformatics/btz067`.

[19] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *CoRR*, abs/1710.09412, 2017. URL `http://arxiv.org/abs/1710.09412`.