Deep Learning Approaches to Antibiotic Classification

Haodong Liu, Jui-Chia Chung, Junda An,Tiancheng Zheng
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We proposed two approaches to perform antibiotic molecule discovery and classification: 1. Simple Graph Neural Network(SGC): Applying graph neural network on atom level features. 2. Fingerprint Vector Model: Applying conventional neural network on molecular level features (molecular fingerprint). We used a dataset of 2335 molecules with labels, and extracted atom/molecular features from the dataset. Baseline models are built for both approaches. Our multi-layer perceptron model applied to molecular fingerprints is able to achieve a 85.2% testing accuracy. On the other hand, our SGC model only reaches an accuracy of 67%. We discovered the reason was that the dataset is unbalanced, with 95% of samples in one label of the two. Since the data is collected by biochemistry experiments, the amount of data is also limited. We then applied techniques, including adjusted weight for labels, Synthetic Minority Oversampling Technique (SMOTE), model ensemble, to try to compensate for the limitation in dataset. Those techniques, along with modification in model architecture significantly improve model performance compared to the midterm report, especially in the fingerprint model. After comparing the performance of the two approaches, we concluded that the fingerprint vector model is more robust than SGC in our experiment.
Data and code are available at our GitHub link:
https://github.com/liujerry1997/Interpretable-activity-prediction.

1 Introduction

Antibiotic is a type of antimicrobial substance that is active in treatment against bacteria. A well know example is penicillin, which is an effective treatment against bacterial infections. Due to the rapid emergence of antibiotic-resistant bacteria, there is a growing need to discover new antibiotics. The development of antibiotics has been an important but difficult problem. The capacity of traditional screening of microbes, whose secondary metabolites inhibit the growth of bacteria, is decreasing as the same molecules are repeatedly discovered (Cox et al., 2017). Screening of chemical libraries is also found to be inefficient and expensive (Brown et al., 2014). New methods to identify antibiotics are necessary.

In this project, we aim to compare two approaches in identifying antibiotics. The first approach is to translate the chemical structure of compounds into fingerprint vectors, which represents the presence of functional groups in the compounds. Machine learning classifiers and deep neural networks are then applied to the fingerprint vectors to classify the compounds. The second approach is to apply graph neural network on atom-level features of compounds to classify if it is a potential antibiotic.

Our goal for this project is to compare the two models performance. Additionally, an interesting characteristic of SGC is, it does not involve non-linearity in the network. Therefore, for future work, it is possible to look into interpretability of our model and identify features or functional groups that strongly correlate with the compounds' antimicrobial properties.

2 Literature Review

The main inspiration of this project is from the paper, "A Deep Learning Approach to Antibiotic Discovery." In the paper, Stokes, et al. (2020) utilized a directed-message passing deep neural network (Yang et al., 2019) to predict antibiotics based on molecule structure. The model has an accuracy of 51.5% and ultimately resulted in identifying halicin, a broad-spectrum bactericidal antibiotic with exceptional in vivo efficacy. However, one concern with this approach is that it functions as a black box, which takes an input and produces a binary output. The middle process is too complicated to provide researcher insight about the question why. However, understanding why a compound is a strong candidate antibiotic is important for downstream experimental validation by chemists. In this work, we examine two alternative approaches that may allow simpler explanations of why a compound is identified as a high-scoring candidate.

Our first approach, Fingerprint Vector Model, uses a fingerprint vector to indicate the presence of substructures, instead of training network with a complete molecule graph structure. Molecule features are extracted and hashed to set the bits of vector, *i.e.* each fingerprint bit represents a fragment of the molecule. Fingerprint vectors has been shown to be useful for classifying antibiotic well, even with simple approach, such as logistic regression. (Information from communicating with a faculty member. Unpublished result.) We want to test the feasibility of other machine learning classifiers and classic neural networks (multi-layer perceptron(MLP) and time-delay neural network(TDNN)) to achieve a higher accuracy.

Our second approach, Simplifying Graph Convolutional Networks proposed by Wu et al. (2019), reduces the excess complexity in graph neural network by removing nonlinearities and collapsing weight matrices between consecutive layers. This simplified method gives us the opportunity to look into weights and nodes model, and possibly interpret important features and substructure. The method requires atom-level features from the molecules. We found another literature by Yang et al. (2019), which gives a detailed illustration of feature extraction method.

We'll compare these two approaches and try to extract important features/substructure that could provide interpretability, in other word, why a graph structure is classified as antibiotic.

3 Contribution and Experiments

3.1 Dataset description

The file for training dataset contains 2,335 rows and 4 columns: "Mean_Inhibition," "SMILES," "Name," and "Activity." Mean inhibition is a score of anti-bactericidal ability. SMILES are the string representations of the molecules' graph structure.

The file for testing dataset contains 162 rows and 6 columns: "Broad_ID," "Name," "SMILES," Pred_Score," "ClinTox (low = less predicted toxicity)," "Mean_Inhibition," and "Activity."

Our training dataset is from the paper "A Deep Learning Approach to Antibiotic Discovery" (Stokes et al., 2020). According to the paper, the authors obtained the 2,335 training data points by "screening for growth inhibition against *E. coli* BW25113 using a widely available US Food and Drug Administration (FDA)-approved drug library consisting of 1,760 molecules" (Stokes et al., 2020). They also included an additional 800 natural products isolated from plant, animal, and microbial sources to further increase chemical diversity, resulting in a total of 2,560 molecules, including 2,335 unique compounds.

3.2 Fingerprint Vector Model

We experimented 4 different kinds of models: a linear perceptron classifier, a support vector machine (SVM) classifier, a multi-layer perceptron (MLP) neural network, and time-delay neural networks (TDNN), one with padding and the other without padding. We hypothesized that a deeper neural network can better capture features in the molecules and achieve a higher accuracy than simple approaches, so the machine learning classifiers are our baseline models.

We follow the literature to preprocess the data, using open source code. First, we convert data containing molecules (SMILES string) into Morgan fingerprints using the RDKit library (Landrum, 2006), and then we convert the fingerprints to feature bit vectors of size 2048 using a radius of 2. In

the end, for each molecule in our data, we have 2048 features and a binary label of 0 and 1, with 1 indicating that the molecule is an active antibiotic and 0 indicating that it is not.

For the traditional machine learning classifiers, we used the implementations from the scikit-learn library with minor parameter tuning. For the MLP, we used PyTorch (Paszke et al., 2019) to implement a neural network with 4 hidden layers, each consisting of a linear layer, a batch normalization layer, a ReLU activation layer, and a dropout layer. The MLP neural network is trained for 100 epochs using the SGD optimizer with momentum (Robbins, 2007), the Cross-Entropy Loss criterion, and the ReduceLROnPlateau learning rate scheduler. The initial learning rate is 1e-3, the weight decay is 5e-6, and the momentum is 0.9. These values are chosen as they are known to help the SGD optimizer converge faster. The learning rate is reduced by a factor of 0.8 once the validation loss stopped improving. We trained for 100 epochs with a batch size of 32.

For the TDNN, we used PyTorch to implement a TDNN with padding and another without padding. The TDNN with padding model consists of 5 1D-convolutional layers (each with a kernel size of 3, a stride of 2, and a padding size of 1), and each followed by a batch normalization layer, a ReLU activation layer and a dropout layer. We also have a max-pooling layer with a kernel size of 2 and a linear layer at the end. The input/output channel sizes are $1/2048 \rightarrow 2048/1024 \rightarrow 1024/512 \rightarrow 512/256 \rightarrow 256/128$. The TDNN without padding model has the same architecture, except that the 1D-convolutional layers have a kernel size of 2, a stride of 2, and no padding. Similar to the MLP model, both TDNN models are trained for 100 epochs using the SGD optimizer with momentum, the Cross-Entropy Loss criterion, and the ReduceLROnPlateau learning rate scheduler. The initial learning rate is 1e-3, the weight decay is 5e-6, and the momentum is 0.9. The learning rate is reduced by a factor of 0.8 once the validation loss stopped improving. We trained both models for 100 epochs with a batch size of 32.

A caveat of this approach is that we are working with preprocessed data, where the molecule features have already been extracted and hashed. Since our model tries to capture information from the preprocessed data, how well the fingerprinting represents the features affects the performance of our models. Another caveat is that the model does not tell which features are important for the antibiotic properties, so further analysis including biological experiments are needed to validate the antibiotic activity of the molecule and identify the correlating important features.

In addition, since the training data is highly unbalanced, with 2215 samples classified as inactive and only 120 samples classified as active, we used two different approaches to make the learning of the models more balanced. The first approach is to put more weight on the class with fewer samples. Specifically, we initialized the weights for both classes to be 1 / *number_of_samples*. The second approach is to use the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al. 2002) and its implementation from the imbalanced-learn Python library to oversample the minority class so that both classes have the same amount of samples.

3.3 Graph Convolution Network

As the Simplifying Graph Convolutional Network (SGC) requires features from atom level, the first step of experiment is to extract those features. From a single molecular SMILES string, RDKit toolkit has functions that could extract atoms features as shown in figure 1 (Yang et al., 2019). All features of an atom are encoded into a feature vector with length of 133, and the molecule is an array of atom-level feature vectors.

feature	description	size
atom type	type of atom (ex. C, N, O), by atomic number	100
# bonds	number of bonds the atom is involved in	6
formal charge	integer electronic charge assigned to atom	5
chirality	unspecified, tetrahedral CW/CCW, or other	4
# Hs	number of bonded hydrogen atoms	5
hybridization	sp, sp2, sp3, sp3d, or sp3d2	5
aromaticity	whether this atom is part of an aromatic system	1
atomic mass	mass of the atom, divided by 100	1

Figure 1: 133 atom features used in Yang et al., 2019 paper

To illustrate the model, here is formal definition from Wu et al., 2019: We define a graph as $G = (V, A)$, in which V represents the molecule's atoms $\{v_1, ..., v_n\}$ and $A \in R^{n \times n}$ is an adjacency matrix where a_{ij} denotes the bond weight between atoms v_i and v_j. $a_{ij} = 0$ if there is no bond between v_i and v_j. We use $D = diag(d_1, ..., d_n)$ to denote a degree matrix where $d_i = \sum_j^n a_{ij}$. Each atom v_i has a $d-$dimensional feature vector $x_i \in R^d$. The feature matrix of an atom is the concatenation of feature vectors of all atoms in it, which is $X = [x_1, ..., x_n]^T$. The label is $Y \in \{0, 1\}$, where 1 denotes that the molecule is an antibiotic and 0 denotes that it is.

Each GCN layer consists of two parts. The first part is the feature aggregation, where feature vector x_i of each atom v_i is averaged with the feature vectors of atoms connected to it.

$$x_i^k = \frac{1}{d_i + 1} x_i^{k-1} + \sum_{j=1}^{n} \frac{a_{ij}}{\sqrt{(d_i + 1)(d_j + 1)}} h_j^{k-1}$$

We can denote the update rule by using matrix notation. $X^k = SX^{k-1}$, where $S = D^{-\frac{1}{2}}(A+I)D^{-\frac{1}{2}}$. This feature aggregation step makes the atom learning from features of its neighbors.

The second step is feature transformation. After the feature aggregation, each layer is associated with a learned weight matrix W^k which transforms the feature linearly. The update rule for this step is:

$$X^k = X^k W^k$$

Unlike GCN, SGC does not have nonlinear layer. Therefore, our assumption is that the nonlinearity between GCN layers is not critical and the major feature is learned through feature aggregation from neighbors. Therefore, the output of SGC with k layers is $Y = softmax(S...SSXW^1W^2...W^k) = softmax(S^kXW)$, where $W = W^1W^2...W^k$.

Our baseline SGC model consists of 6 GCN layers and apply softmax on the output.

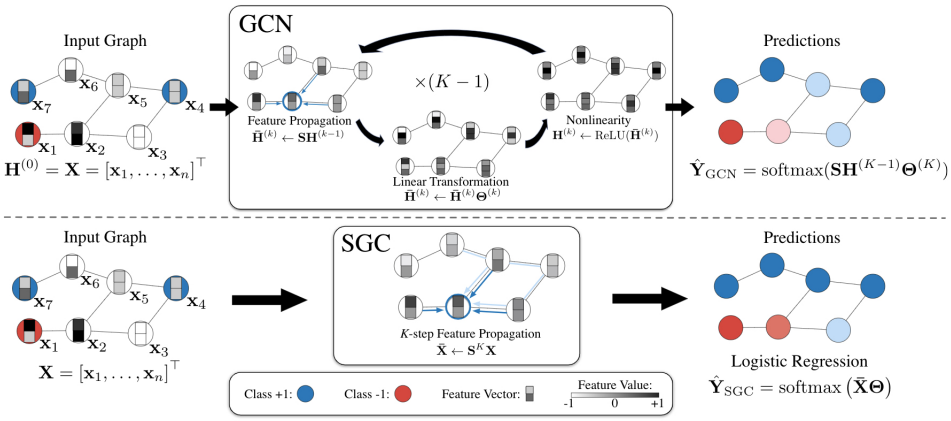


Figure 2: GCN vs. SGC from Wu et al., 2019.

Our model utilized some functions from open-source from Stokes, et al., 2020 and Wu et al., 2019. Besides baseline, we tried to adjust the parameters of the network, and implemented model ensemble and hybrid model as suggested by Stokes, et al. (2020). The motivation and explanation for the implementation is below.

Model ensemble: We have a small and biased dataset, with 95% of samples belongs to one of the binary labels. We tried to balance the data feed into the network, so the network does not naively predict all samples as the major label. A solution is training multiple models, each with balanced number of positive and negative sample. For number of ensemble, we used 20 as suggested by Stokes, et al. (2020). For each ensemble training, we prepared the dataset using all 100 positive samples (5% of the data), and randomly pick 200 samples from 1768 negative samples; We initialize the model with different initialization weight. During testing, all model's output vote to decide the final output. When tuning for the model, We adjust those parameters:
1. Ensemble number.
2. Positive and negative sample ratio for each ensemble training.
3. Propagation degree (number of GCN layers)
4. Method to condense atom features to whole molecule feature (mean, sum, node selection)
5. Learning rate
6. Scheduler factor.

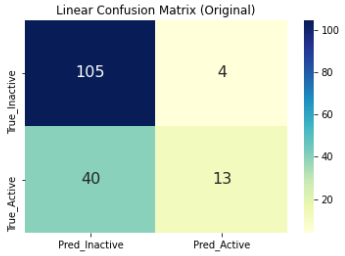
Hybrid Model: The baseline SGC model utilizes atom features to make prediction. The last linear layer takes the sum of all atom's 133 features, as a 133 * 1 vector. We propose adding 2d-normalized molecular feature extracted using RDKit package (Landrum, 2006), a 200 $\times$ 1 vector, before the last linear layer. We hypothesized this additional information would help the model to learn a better representation of the data.

4 Evaluation metric and Result

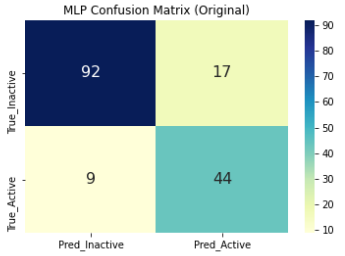
4.1 Fingerprint Vector Models

4.1.1 Fingerprint Vector Models with Original Data

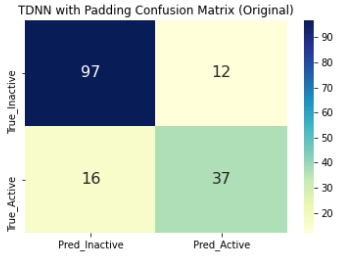
The confusion matrices of testing with original imbalanced data without adjusting weights and performing SMOTE are shown in Figure 3. Confusion matrix for SVM with original data is shown in appendix (Figure 15a). The training and validation losses and accuracies of MLP and TDNN are shown in appendix (Figure 12).



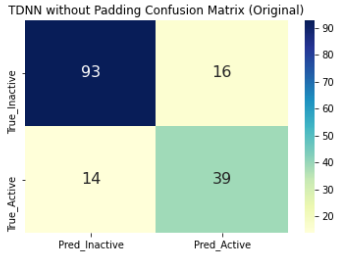
(a) Linear Perceptron



(b) MLP



(c) TDNN (with padding)



(d) TDNN (without padding)

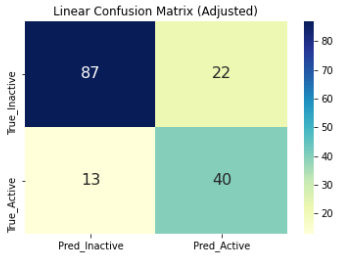
Figure 3: Confusion matrices (Original data)

For the fingerprint vector models, we used two different evaluation metrics: the testing accuracy and the AUC score. From the results in the table we can see that our MLP classifier and TDNN classifiers have the best performance in terms of testing accuracy and AUC score. SVM and linear perceptron models have the worst performance with the lowest testing accuracy and AUC scores. This result is consistent with our hypothesis that deep neural networks can capture complex feature patterns better than traditional machine learning models.

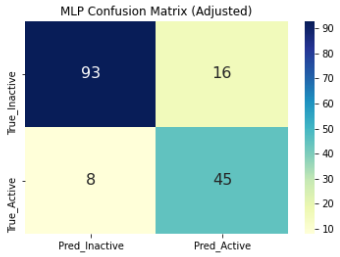
	Linear Perceptron Classifier	SVM Classifier	MLP Classifier	TDNN (padding)	TDNN (no padding)
Training Accuracy	0.997	0.979	1.000	0.999	0.998
Validation Accuracy	0.968	0.968	0.979	0.970	0.976
Testing Accuracy	0.728	0.728	0.840	0.827	0.815
Testing AUC Score	0.604	0.604	0.837	0.794	0.795

4.1.2 Fingerprint Vector Models with Adjusted Weight data

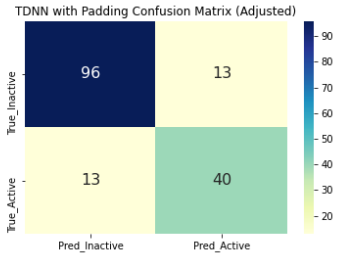
To deal with imbalance data, we tried to applied weights to each class label as our tutor Jacob suggested. The confusion matrices of testing with adjusted weights are shown in Figure 4. Confusion matrix for SVM with adjusted weights is shown in appendix (Figure 15b). The training and validation losses and accuracies of MLP and TDNN are shown in appendix (Figure 13).



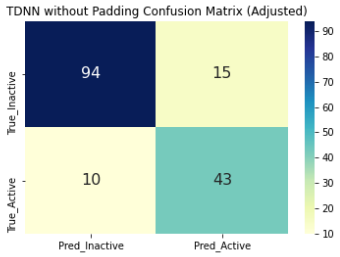
(a) Linear Perceptron



(b) MLP



(c) TDNN (with padding)



(d) TDNN (without padding)

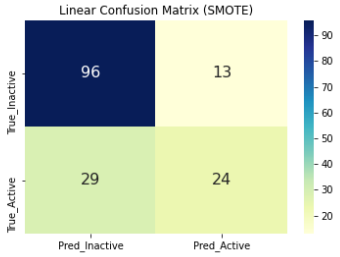
Figure 4: Confusion matrices (Adjusted weights)

From the table, we can see a significant improvement in the linear perceptron classifier and the SVM classifier.

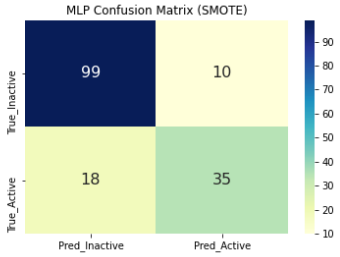
	Linear Perceptron Classifier	SVM Classifier	MLP Classifier	TDNN (padding)	TDNN (no padding)
Training Accuracy	0.994	0.995	0.999	0.995	0.996
Validation Accuracy	0.944	0.944	0.968	0.972	0.966
Testing Accuracy	0.784	0.784	0.852	0.840	0.846
Testing AUC Score	0.776	0.776	0.851	0.818	0.837

4.1.3 Fingerprint Vector Models with SMOTE data

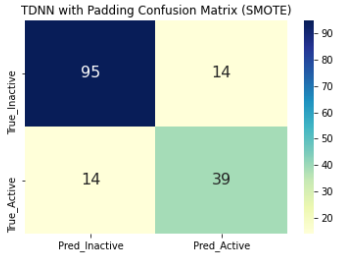
To deal with imbalance data, we also tried to use the imbalanced-learn Python library, specifically the Synthetic Minority Oversampling Technique (SMOTE) (Chawla et al. 2002), to oversample the minority class so that both classes have the same amount of samples. The confusion matrices of testing with SMOTE are shown in Figure 5. Confusion matrix for SVM with SMOTE is shown in appendix (Figure 15c). The training and validation losses and accuracies of MLP and TDNN are shown in appendix (Figure 14).



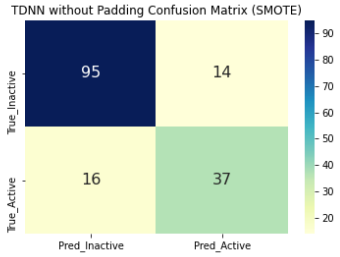
(a) Linear Perceptron



(b) MLP



(c) TDNN (with padding)



(d) TDNN (without padding)

Figure 5: Confusion matrices (SMOTE data)

	Linear Perceptron Classifier	SVM Classifier	MLP Classifier	TDNN (padding)	TDNN (no padding)
Training Accuracy	0.998	0.998	1.000	1.000	1.000
Validation Accuracy	0.988	0.988	0.993	0.988	0.985
Testing Accuracy	0.741	0.741	0.827	0.827	0.815
Testing AUC Score	0.667	0.667	0.784	0.804	0.784

4.1.4 Findings

As we can see from the figures and charts, the performance of deep learning models, both MLP and TDNN, show great robustness towards imbalanced data and have similar performance throughout the three experiments. Adjusting the class weights slightly improves the performance of the deep learning models. On the other hand, the performance of traditional machine learning models depends greatly on the distribution of the training data. They exhibit poor performances on the original data, classifying most of the testing data as inactive, which is the majority class. However, after using adjusted weights and SMOTE, their performances improved greatly. Most notably, when using adjusted class weights, their performances are almost comparable to that of deep learning models.

4.2 Simplifying Graph Convolutional Network

Below are the results model obtained from parameter:
1. Ensemble number: 20
2. Positive and negative sample ratio for each ensemble training: 1:2
3. Propagation degree (number of GCN layers): 6
4. Method to condense atom features to whole molecule feature: Summation
5. Scheduler factor: 0.8
6. Training epoch: 50

	SGC model	SGC model with ensemble	SGC model with ensemble & molecular features
Training Accuracy	0.943	0.585	0.72
Validation Accuracy	0.957	0.748	0.860
Testing Accuracy	0.673	0.64	0.65
Testing AUC Score	0.461	0.473	0.45

After trying adjusting parameters for SGC and ensemble and hybrid method, we realize that SGC is not able to extract useful information from atom's sparse features vector in our dataset. It tends to predict all sample into negative, which is the majority label in the dataset. This is the reason for high accuracy for training and validation, but low accuracy in testing, as the testing dataset includes a more balanced number of positive and negative samples.

SGC model result:

(a) SGC Train/Valid accuracy Plot (b) SGC Train/Valid Loss Plot

Figure 6: Result figure for SGC

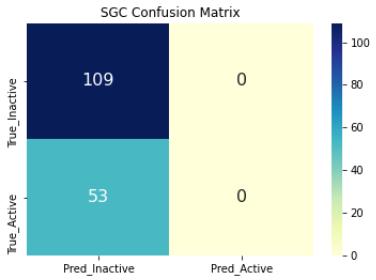
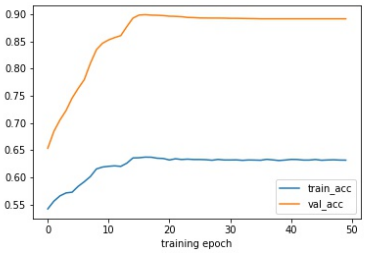
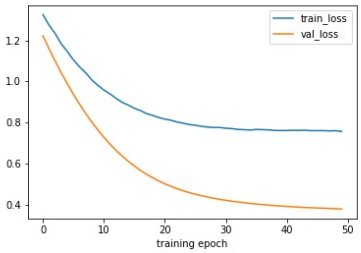


Figure 7: SGC confusion matrix

Result for SGC model with ensemble: While in the ensemble cases, the number of sample (300 samples in 1:2 ratio) feed into network is too little for network to learn. As shown in figure below, the model's performance very similar to SGC model without ensemble.



(a) SGC ensemble Train/Valid accuracy Plot



(b) SGC ensemble Train/Valid loss Plot

Figure 8: Result figure for SGC ensemble model

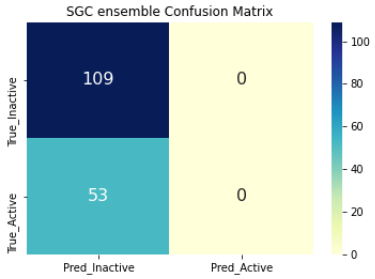
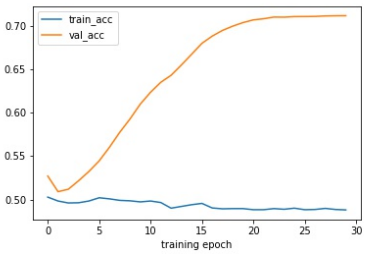
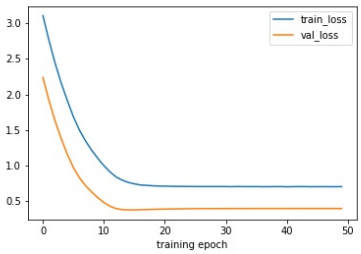


Figure 9: SGC ensmeble confusion matrix

Result for SGC model with ensemble and molecular feature: Adding molecular feature to the network does not improve the model either. The model does not learn generalized representation of the data. Similarly, it only outputs the popular label, which is negative.



(a) SGC ensemble with molecular feature Train/Valid accuracy Plot



(b) SGC ensemble with molecular feature Train/Valid Loss Plot

Figure 10: Result figure for SGC ensemble model with molecular feature

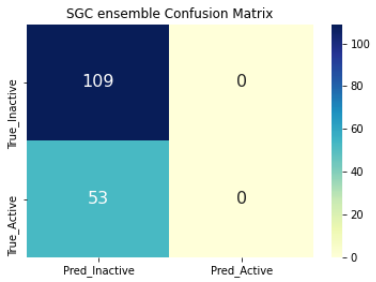


Figure 11: SGC ensmeble with molecular feature confusion matrix

Here is a list of parameters we attempted. However, none of those improve the model significantly.
1. Ensemble number: 5 - 20
2. Positive and negative sample ratio for each ensemble training: 1:1, 1:2, 1:10
3. Propagation degree (number of GCN layers): 5, 6, 7, 8
4. Method to condense atom features to whole molecule feature: mean, sum
5. Scheduler factor: 0.1-0.8
6. Adjusted weight for label

As mentioned earlier, we might be able to extract interpretability of the model from the weight matrix of the final linear layer. However, as the performance of SGC indicates, it cannot provide insight to the molecular feature at this moment.

5 Conclusion and Future Work

In this antibiotics classification task, we achieve a performance of 84.0% testing accuracy, using fingerprint vectors with the MLP model. Simple machine learning classifiers have relatively poorer performance. Therefore, the Fingerprint Vector with MLP model approach could be useful for classifying antibiotics. However, it may not be able to tell what molecular features are more important for the antimicrobial property.

As we achieve high accuracy in the Fingerprint Vector Models, we looked deeper into the dataset and found out that the data is unbalanced, with more than 95% of the samples belonging to one of the two classes. This problem is then tackled with adjusted weights and SMOTE. In particular, weight adjustment is able to boost the testing accuracy of the MLP model to 85.2%.

We applied data augmentation techniques such as model ensemble, and hybrid model in SGC model. After hyperparameter adjustment and comparing two model's performance, we concluded fingerprint vector model is more suitable than SGC in our experiment.

Here is some future direction for for SGC model:
1. Since the SGC layer only takes the average or sum of the features of neighbors, the bond type is not taken into account. we could experiment assigning different weights to different bonds to improve the SGC model . There are four types of bonds in molecule, which are single bond, double bond, triple bond, and aromatic bond. Our guess is that the feature aggregation step might benefit from the bond information, which will allow the model to learn more features.
2. Given that directly taking the sum or the mean of the final atom features does not generate good results, it might be a good practice to create a master node that can accumulate features from all atoms in the message passing process. The master is used to predict whether the molecule is an antibiotic or not. 3. We'll also try to increase the dataset size by following work done by (Stokes, et al., 2020), to see if there is more data available.
4. Another possible way to improve the SGC model is to embed atom features in a lower dimension. Since the original atom feature is a one hot vector which is too sparse, the model cannot learn well from it. Therefore, embedding the sparse feature vectors to a lower dimension, just like word embedding, might generate a better result.
5. Finally, after we get a better SGC model, we want to explore the intrepretability of the model,

and identify features or functional groups that strongly correlate with the compounds' antimicrobial properties.

6 Division of work among team members

Fingerprint Vector Models approach: Jui-Chia and Tiancheng
Simple GNN approach: Junda and Haodong

7 References

Brown, D. G., May-Dracka, T. L., Gagnon, M. M., & Tommasi, R. (2014). Trends and exceptions of physical properties on antibacterial activity for Gram-positive and Gram-negative pathogens. Journal of medicinal chemistry, 57(23), 10144-10161.

Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: synthetic minority over-sampling technique. Journal of artificial intelligence research, 16, 321-357.

Cox, G., Sieron, A., King, A. M., De Pascale, G., Pawlowski, A. C., Koteva, K., & Wright, G. D. (2017). A common platform for antibiotic dereplication and adjuvant discovery. Cell chemical biology, 24(1), 98-109.

Landrum, G. (2006). RDKit: Open-source cheminformatics.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., ... & Desmaison, A. (2019). Pytorch: An imperative style, high-performance deep learning library. In Advances in neural information processing systems (pp. 8026-8037).

Robbins, H. (2007). A Stochastic Approximation Method. Annals of Mathematical Statistics, 22, 400-407.

Stokes, J. M., Yang, K., Swanson, K., Jin, W., Cubillos-Ruiz, A., Donghia, N. M., ... & Tran, V. M. (2020). A deep learning approach to antibiotic discovery. Cell, 180(4), 688-702.

Wu, F., Zhang, T., Souza Jr, A. H. D., Fifty, C., Yu, T., & Weinberger, K. Q. (2019). Simplifying graph convolutional networks. arXiv preprint arXiv:1902.07153.

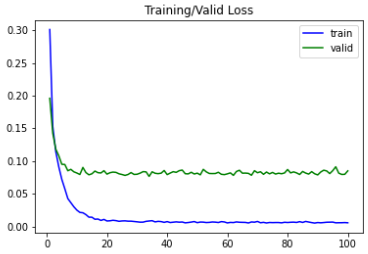
Yang, K., Swanson, K., Jin, W., Coley, C., Eiden, P., Gao, H., ... & Palmer, A. (2019). Analyzing learned molecular representations for property prediction. Journal of chemical information and modeling, 59(8), 3370-3388.

Zampieri, M., Zimmermann, M., Claassen, M., & Sauer, U. (2017). Nontargeted metabolomics reveals the multilevel response to antibiotic perturbations. Cell reports, 19(6), 1214-1228.

8 Appendix

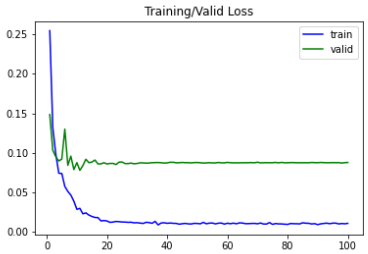
All MLP and TDNN models are trained with 100 epochs and batch size of 32.

Figure 12a and 12b shows the training and validation loss and accuracy of MLP, using original data. Figure 12c and 12d shows the training and validation loss and accuracy of TDNN with padding, using original data. Figure 12e and 12f shows the training and validation loss and accuracy of TDNN without padding, using original data.

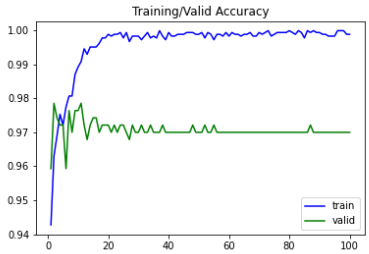


(a) MLP Training and Validation Loss vs. epoch number

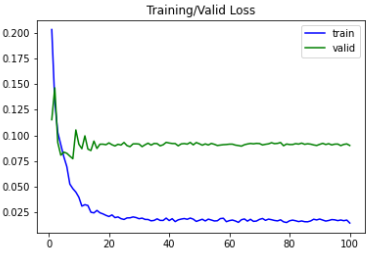
(b) MLP Training and Validation Accuracy vs. epoch number



(c) TDNN (with padding) Training and Validation Loss vs. epoch number



(d) TDNN (with padding) Training and Validation Accuracy vs. epoch number



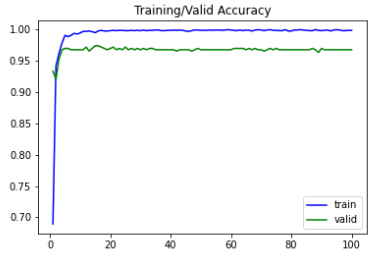
(e) TDNN (without padding) Training and Validation Loss vs. epoch number

(f) TDNN (without padding) Training and Validation Accuracy vs. epoch number

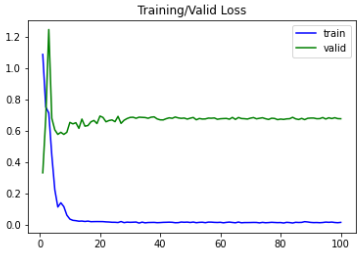
Figure 12: Training and Validation Loss and Accuracy Plots for MLP and TDNN Using Original Data

Figure 13a and 13b shows the training and validation loss and accuracy of MLP, with adjusted weights. Figure 13c and 13d shows the training and validation loss and accuracy of TDNN with padding, with adjusted weights. Figure 13e and 13f shows the training and validation loss and accuracy of TDNN without padding, with adjusted weights.

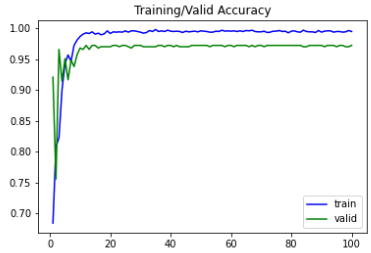
(a) MLP Training and Validation Loss vs. epoch number



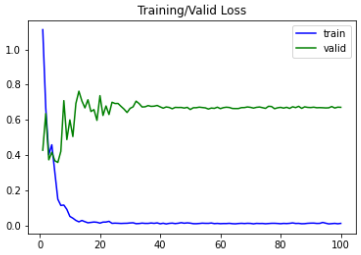
(b) MLP Training and Validation Accuracy vs. epoch number



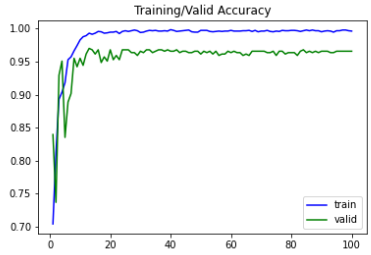
(c) TDNN (with padding) Training and Validation Loss vs. epoch number



(d) TDNN (with padding) Training and Validation Accuracy vs. epoch number



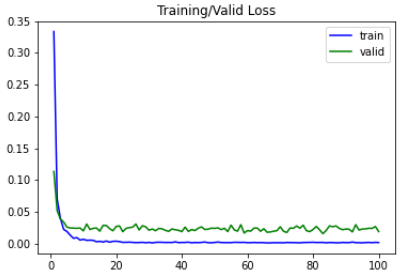
(e) TDNN (without padding) Training and Validation Loss vs. epoch number



(f) TDNN (without padding) Training and Validation Accuracy vs. epoch number

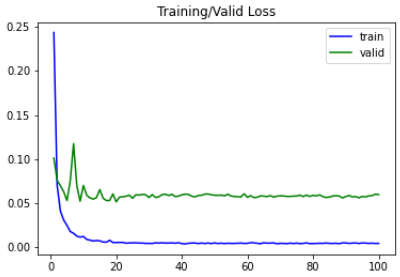
Figure 13: Training and Validation Loss and Accuracy Plots for MLP and TDNN with Adjusted Weights

Figure 14a and 14b shows the training and validation loss and accuracy of MLP, using SMOTE. Figure 14c and 14d shows the training and validation loss and accuracy of TDNN with padding, using SMOTE. Figure 14e and 14f shows the training and validation loss and accuracy of TDNN without padding, using SMOTE.

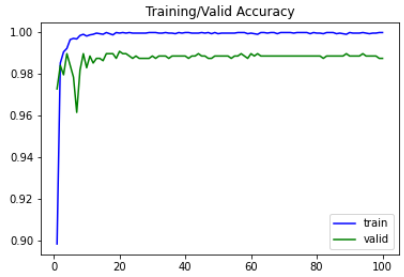


(a) MLP Training and Validation Loss vs. epoch number

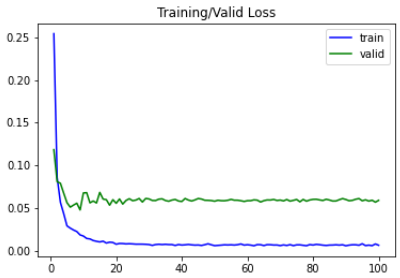
(b) MLP Training and Validation Accuracy vs. epoch number



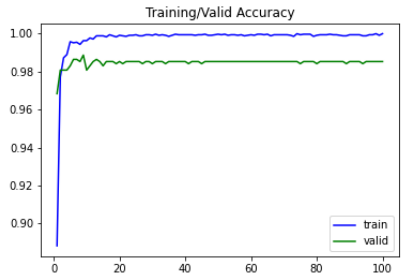
(c) TDNN (with padding) Training and Validation Loss vs. epoch number



(d) TDNN (with padding) Training and Validation Accuracy vs. epoch number



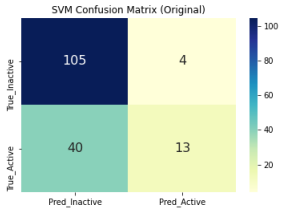
(e) TDNN (without padding) Training and Validation Loss vs. epoch number



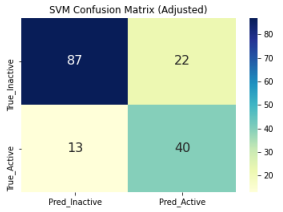
(f) TDNN (without padding) Training and Validation Accuracy vs. epoch number

Figure 14: Training and Validation Loss and Accuracy Plots for MLP and TDNN Using SMOTE

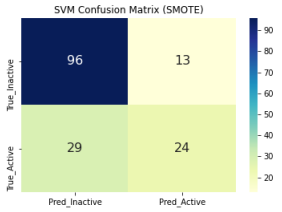
Figure 15 shows the SVM confusion matrices using original data (a), adjusted weights (b), and SMOTE (c).



(a) SVM Confusion Matrix (Original Data)



(b) SVM Confusion Matrix (Adjusted Weights)



(c) SVM Confusion Matrix (SMOTE)

Figure 15: SVM Confusion Matrices