AI-guided detection of antibiotic-resistant bacteria using resistance genes

Understanding the importance of pre-training and model complexity when using transformer-based techniques to detect antibiotic resistance

Master's thesis in Biomedical Engineering

Erik Aerts

DEPARTMENT OF MATHEMATICAL SCIENCES

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis 2024

AI-guided detection of antibiotic-resistant bacteria using resistance genes

Understanding the importance of pre-training and model complexity when using transformer-based techniques to detect antibiotic resistance

ERIK AERTS

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Applied Mathematics and Statistics
Erik Kristiansson group
Chalmers University of Technology
Gothenburg, Sweden 2024

AI-guided detection of antibiotic-resistant bacteria using resistance genes
Understanding the importance of pre-training and model complexity when using transformer-based techniques to detect antibiotic resistance
ERIK AERTS

© ERIK AERTS, 2024.

Supervisor: Erik Kristiansson & Anna Johnning, Mathematical Sciences
Examiner: Erik Kristiansson, Mathematical Sciences

Master's Thesis 2024
Department of Mathematical Sciences
Applied Mathematics and Statistics
Erik Kristiansson group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2024

AI-guided detection of antibiotic-resistant bacteria using resistance genes
Understanding the importance of pre-training and model complexity when using transformer-based techniques to detect antibiotic resistance
Erik Aerts
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Antibiotic resistance is threatening advancements made in modern medicine. Understanding the genomics behind multi-resistant profiles can assist in planning the correct treatment which can lower the abundance of antibiotic usage and hamper the vicious resistance cycle. Transformer-based AI models have shown state-of-the-art performance in understanding complex patterns in data. The thesis aimed to create a framework on how to implement transformers to predict bacterial resistance profiles by training on genomic data. The framework consisted of a transformer-based encoder and parallel classification networks for predicting antibiotic susceptibility. Each model trained on antibiotic resistance genes (ARGs) from *Escherichia coli* where a subset of isolates had recorded resistance profiles.

The results showed that having a high complexity in the encoder is key for the model to accurately predict resistance to antibiotics where the occurrence of resistance is rare. This is relevant for any clinical setting, as models with less than 12 encoder blocks could not find these resistance profiles. The framework benefited from pre-training on unlabeled genomic data as performance generally increased. However, the type of masked language model pre-training which benefited the system more was situational and no conclusion was drawn. Finally, the thesis also found features in the data on which the models were basing decisions off on. The number of ARGs of an isolate was deemed the most influential feature in the data which relates to how much information the transformer can process. Following, relations between ARGs *gyrA-D87N / parC-S80I* and *aph(3")-Ib / aph(6')-Id* were shown to be an important decision basis for the models. Likewise, two point mutations of the *pmrB* gene also stood out as important ARGs in the decision-making processes for the models. The reasons why these ARGs are weighted highly by the models are currently unknown but are of interest to be studied further for a better understanding of underlying factors to multi-resistance.

Keywords: artificial intelligence, antibiotic resistance, transformer, self-attention, embedding, encoder, pre-training, fine-tuning, masked language modelling.

Acknowledgements

I would like to thank the Department of Mathematical Sciences for their warm welcoming of me, and a special thank you to those involved with my project. To Erik and Anna, thank you for your time, patience and guidance throughout the spring. The motivation you have given me has been instrumental in the enjoyment of the project. A special thank you to my colleagues, Jesper and Michaela, for all the help and discussions you have provided and lastly, I would like to thank family and close friends for their encouragement and support.

Erik Aerts, Gothenburg, June 2024

List of Acronyms

Below is the list of acronyms- and class for each antibiotics used in the thesis, listed in alphabetical order:

Acronym	*Name*	*Class*
AMC	Amoxicillin + clavulanic acid	Penicillins
AMP	Ampicillin	Penicillins
AZT	Aztreonam	Monobactams
CZL	Cefazolin	Cephalosporins
CPM	Cefepim	Cephalosporins
CTX	Cefotaxime	Cephalosporins
CXT	Cefoxitin	Cephalosporins
CPD	Cefpodoxime	Cephalosporins
CTZ	Ceftazidime	Cephalosporins
CTR	Ceftriaxone	Cephalosporins
CMP	Chloramphenicol	Others
CIP	Ciprofloxacin	Quinolones
ETP	Ertapenem	Carbapenems
GEN	Gentamicin	Aminoglycosides
IMI	Imipenem	Carbapenems
LVX	Levofloxacin	Quinolones
MEM	Meropenem	Carbapenems
STR	Streptomycin	Aminoglycosides
SXT	Sulfamethoxazole	Sulfonamides
INN	Sulfisoxazole	Sulfonamides
TET	Tetracycline	Tetracyclines
TOB	Tobramycin	Aminoglycosides
TRI	Trimethoprim	Others
TMP	Trimethoprim + sulfamethoxazole	Aminoglycosides

Contents

List of Figures

List of Tables

1
Introduction

Since the start of the 20th century, human life expectancy has seen a rapid increase. In fact, between the years 1900- and 2021, the expected lifespan of a newborn has increased by 122%: from 32 years to 71 years [1]. One of many important factors in this positive change has been the health care's ability to consistently treat common diseases associated with bacterial infections with the use of antibiotic substances [2]. After the discovery by Sir Alexander Fleming in 1928, the first antibiotic substance was commercially available on the market in 1945. This marked the start of the golden era of the discovery of novel antibiotics, with the last new class of antibiotics being discovered in the 1970s at the current time of writing [3]. Ever since, the demand for antibiotic substances has continued to grow to maintain the continuous growth in health standards. As a reference, the global consumption of antibiotics has seen an increase of 48% since the start of the millennium [4].

However, with the abundance of usage, a global increase in bacterial antibiotic resistance (AR) has been observed ever since the substances were introduced as a treatment option. The first documented signs of antibiotic resistance date back to 1942 when strains of *Staphylococcus aureus* were found to resist the action of penicillin in hospitalized patients during trials. The rate of resistance continued to grow yet was not deemed as problematic during the golden era of the discovery of novel antibiotics since any discovered resistance was combated with a newly developed class of antibiotics. However, the ability to develop new classes of antibiotics has not continued to limit the growth of AR [5]. Bacterial pathogens with the ability to resist conventional treatment alternatives challenge many of the advancements made in modern medicine [6]. It was estimated that AR contributed to 4.95 million deaths worldwide in 2019 and was deemed the core reason for 1.27 million of those deaths [7]. Moreover, antibiotic substances play an important role in a variety of medical treatments- and procedures not directly associated with combating an ongoing infection including areas such as surgery, cesarean sections and cancer chemotherapy. This increases the vulnerability of the patient when undergoing these procedures, as post-treatment infections may be as difficult to treat as the reason for the procedure itself [6]. In addition to the burden AR has on the healthcare system, The World Bank estimates that AR could result in $1 trillion in additional healthcare costs by 2050 and between $ 1-3.4 trillion GDP losses per year by 2030 [8].

Bacteria can acquire AR by gaining antibiotic resistance genes (ARGs) which encodes for resistance mechanisms needed to withstand the presence of the substances. This includes, but is not limited to, changed permeability in the outer membrane,

developed efflux pumps to expulse specific substances and enzymatic modification or degradation. Importantly, these resistance mechanisms do not have to be substance-specific but can change the susceptibility of a pathogen to a wide range of antibiotics [9, 10]. ARGs can be obtained through mutations in an individual genome or by receiving transferred genetic material, such as plasmids, from its surroundings [11]. These mobile genetic elements can contain a large series of ARGs creating a possible multiresistent profile to different classes of antibiotics. This complicates the decision of which antibiotic substance to use during treatment as the resistance profile of the bacteria in question is usually not known to the physician, thus leading to an increase in antibiotic usage to find a treatment alternative. This may lead to a vicious cycle further decreasing our ability to treat what has recently been treatable [12].

The resistance profile of bacteria can be determined by cultivation in the presence of different antibiotics to determine the degree of susceptibility. The methodology of this test produces well-interpretative results but requires time and laboratory precision to not contaminate tests [13]. Alternatively, the genetic information of the bacteria could instead be studied to find the presence of ARGs and mutations commonly associated with antibiotic resistance. PCR can perform this task whilst being a fast and cheap alternative but is limited by the number of genes it can test per run making the full resistance profile difficult to find [14]. Furthermore, sequencing the entire genome can give the resistance profile of the individual but is a more time-consuming and expensive method [15].

Deep learning is a field within artificial intelligence (AI) that has seen a recent increase in attention from the public after many open-source projects has been made available. Understanding- and communicating with human languages, recognizing- and analyzing features in images and creating media from scratch are a few examples of tasks previously not associated with machines. This has been made possible due to recent advancements in the field, such as the introduction of transformers. The ability to understand complex patterns by these networks has promoted the question regarding if, and how, AI can be integrated into our health-care system. Networks aimed to assist in diagnosis, suggest treatment plans and guide physical robots in surgical procedures are all examples of AI in healthcare with promising results [16].

1.1 Aim and Scope

The thesis aims to create an AI model framework using state-of-the-art deep learning techniques to predict resistance to commonly used antibiotics from genetic data. This framework is supposed to contribute knowledge on important factors when integrating transformers into the field of AI-guided antibiotic resistance predictions. This includes architectural aspects of how performance is related to model complexity and embedding size, to aspects of how masked language training with genomic data affects the ability to make accurate predictions. Moreover, the thesis aims to investigate if self-supervised pre-training can improve performance, and if so, how to make efficient use of the available data. Furthermore, the thesis aims to study the decision process by different models to understand the data patterns the models has found. With this, discovered features and relations in the genomic data aim to increase the knowledge of underlying factors of antibiotic resistance.

The thesis will be limited to only study genomic data from the species *Escherichia coli* as data from other species will be excluded. Furthermore, the thesis will limit the number of studied antibiotics to 24. This means that all models will only predict resistance to the included antibiotics in the study. Since the thesis aims to create a framework to integrate transformer-based models into the field of antibiotic resistance predictions, no other types of deep learning architectures will be studied.

2
Background

Artificial Intelligence (AI) is a field within computer science using mathematical modelling to learn structural patterns with the data without guidance- or interference from an external source. The experience the model has gained from training on a data set can be used to influence decisions the model makes when presented with new unseen information. The subject of AI and machine learning (ML) dates back to 1943 when published work featured a simplistic mathematical model that resembled the human neurons which was able to process boolean information to decide when to activate [17]. Due to restrictions in both hardware and available data, the field of AI saw limited interest levels from both academia and the public for the remaining decades of the 20th century [18]. With the introduction of graphical processing units (GPUs) and better data storage options around the turn of the millennium, the field of AI saw a resurgence. More complex networks were now able to be trained on larger sets of data which began the era of deep learning (DL) [17].

Teaching DL networks can primarily be done with two techniques depending on the task the network is aimed to learn. When giving a network a large quantity of data without additional information the network can cluster similar data together from found patterns which is called unsupervised learning. If the network is aimed to classify data to a subset of predetermined classes, either on a binary- or multi-class level, the use of additional labels to the training data can help teach the network which features to associate with certain labels, also called supervised learning. During training, the network needs some feedback on the performance of the desired task to improve. The use of a loss function to determine a numeric estimate of performance is commonly used in ML. The calculated loss can then be used to determine gradients to be back-propagated through the system to tune the trainable parameters in the system to better fit the data. This parameter optimization process is done with an optimization algorithm, commonly selected to be stochastic gradient descent or adaptive moment estimation (Adam). During training, the network is constantly validating the performance on a subset of the data not available when tuning the parameters. This type of validation is done to ensure that the model is robust enough to process new data and does not overfit on specific patterns found in the training data [19].

The architectural structure of the network is dependent on the task. In the field of natural language processing (NLP), a sub-field of computer science where the aim is for computers to understand and utilize human language, the data is often structured as position-dependent sequences. Recurrent neural networks (RNN) were

commonly used to solve NLP tasks for many years due to their ability to capture sequential dependencies. However, RNNs were often not robust enough to capture long-range dependencies in sequences and had to be implemented with additional features to overcome the problem of vanishing gradients during back-propagation. Moreover, the recurrent nature of the network made parallel calculations of multiple parts of the data impossible which slowed down training. These two limitations for the RNN motivated the development of a new model architecture not dependent on the recurrent nature previously used for sequential data: the transformer [20]. The transformer uses a self-attention mechanism to compute weighted representations for each position to understand contextual relationships in a sequence. The model is also not dependent on the positions in a sequence compared to the RNN architectures. Instead, the full sequence is fed as the input to the network and additional information such as position, order and more are saved as separate encodings. This means that the transformer model is not limited by the same parallelization constraints as the RNN networks [21].

When training a transformer-based model for an NLP task it is common to use Masked Language Modeling (MLM), a self-supervised pre-training technique before adapting the network to a specific aim. This self-supervised learning is performed by hiding away a subset of the data to create self-imposed multiclass-classification problems [22]. This is done by masking out parts of the sequences and then asking the network what was hidden from it. This aims to give the network a better understanding of the language as a whole before the model is adapted to a specific task via fine-tuning techniques. The masking of the data can be changed for each iteration of the entire data set, denoted an epoch, whilst keeping a similar percentage of the number of masked words or having the masking remain constant during the entire training phase. This difference in masking techniques is called dynamic- or static masking respectively [23].

With the state-of-the-art performance in many areas of NLP, the idea of trying transformer-based models and applying the self-attention mechanisms in other areas were not far off [24]. Transformers have been applied in computer vision to perform tasks such as image recognition, semantic segmentation and object detection and have shown outstanding, and sometimes, state-of-the-art performance [25]. In a similar fashion to the sequential nature of the data, transformers were introduced to perform tasks within the fields of chemistry and biology. This has seen a variety of successful applications, such as Chemformer which works to perform sequence-to-sequence and discriminative cheminformatics tasks and AlphaFold which can predict 3D- structures of proteins from amino acid sequences [24, 26].

3
Theory

The following chapter will provide the theoretical aspects needed to understand the methodology and results of the thesis.

3.1 Tokenization & Embedding space

For the model to be able to understand a sequence of ARGs, the input sequence is transformed from a series of letter representations to a vectorized format. The first step in this process is called tokenization of the data where each ARG in the input sequence of length n is represented as a token from a vocabulary created with the N unique elements in the data set. Each n elements in the sequence is converted to a one-hot encoding vector of length N where the position of the given in the vocabulary is set to 1 and every other position is set to 0. With this, all one-hot encoding vectors are concatenated with a sequential start vector creating a one-hot encoding matrix $M \in \mathbb{R}^{N \times n+1}$, which can be expressed as follows [2].

$$M = \begin{bmatrix} 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \\ 0 \end{bmatrix} \overset{\text{concatenated}}{\Big\|} \begin{bmatrix} 0 & 0 & \dots & 0 & 0 \\ 0 & 0 & & 1 & 0 \\ 0 & 0 & & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & & 0 & 0 \\ 1 & 0 & & 0 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}_{N \times n+1}$$

The sequential start vector is regarded as a special token in the system, often denoted as the *CLS*-token. The *CLS*-token marks the start of a sequence and is used for classification tasks [27]. The one-hot encoding matrix is then transformed into an embedding space. Each vector in the one-hot encoding matrix is mapped to a vector of length d_E which is the predetermined dimension of the embedding space. Unlike the fixed nature of the one-hot encoding matrix, the embedding parameters are aimed to be updated to achieve the goal of representing the contextual meaning of the tokens in the sequence [2].

3.2 Self-attention

Self-attention uses attention functions to calculate weighted averages to gain contextual information from the token embeddings. The attention function computes

an output sequence $Y^A = (y_1, ..., y_{n+1})$, of the same length $n + 1$ as the embedded input sequence $X^E = (x_1, ..., x_{n+1})$ by using a set of linear transformations to map the two. The linear transformation is based on the matrices Q containing *n+1* query vectors specific for each token, K containing the *n+1* key vectors used as the contextual reference points and V to compute the desired weighted average of our output. The matrices Q, K and V can be can be calculated as follows, presented in Equation 3.1,

$$\begin{aligned} Q &= W_Q X^E, \\ K &= W_K X^E, \\ V &= W_V X^E \end{aligned} \tag{3.1}$$

where W_Q, W_K and W_V are weight matrices with trainable parameters. The output sequence Y^A from the attention function can be determined as follows, presented in Equation 3.2,

$$Y^A = h_{sm}\left(\frac{QK^T}{\sqrt{d}}\right)V \tag{3.2}$$

where $\sqrt{d}$ is a scaling factor dependent on the dimension of the weight matrices W_Q, W_K and W_V introduced to prevent vanishing gradients in the system [28]. Equation 3.2 contains the activation function h_{sm} which is a softmax operation on the scaled inner-product between Q and K^T. The softmax activation function is defined as follows, see Equation 3.3,

$$h_{sm}(p_i) = \frac{e^{p_i}}{\sum_{k=1}^{m} e^{p_k}} \tag{3.3}$$

for a given set $p_i \in [1, ..., m]$ [2].

3.3 Multi-head attention

When using self-attention techniques to learn contextual information in a transformer model, a common approach is to linearly project the querys keys and values m different times in so called attention heads. The multi-head attention approach can project the input into multiple different subspaces simultaneously and can therefore compute multiple attention functions independently of one another to get a better understanding of the dependencies of the data. The heads work as parallel self-attention layers on the same embedded input, each generating an updated attention embedding Y_i^A $i \in [1, ..., m]$, which is concatenated into a collective output $Y^{A*} \in \mathbb{R}^{Nm \times n+1}$ from the parallel layers [29].

The dimension of the concatenated output Y^{A*} is reduced through a linear transformation with weight matrix W^O with trainable parameters to produce the output of the multi-head attention layer $\hat{Y}^A \in \mathbb{R}^{N \times n+1}$. The dimensions of the output after the linear transformation with W^O equals the embedded input. This dimensional consistency is necessary for information to be passed through multilayered networks, see Equation 3.4 [29].

$$\hat{Y}^A = W^O Y^{A*} = W^O \begin{bmatrix} Y_1^A \\ Y_2^A \\ \vdots \\ Y_{m-1}^A \\ Y_m^A \end{bmatrix} \tag{3.4}$$

A graphical illustration of the multi-head attention process is presented as Figure 3.1.

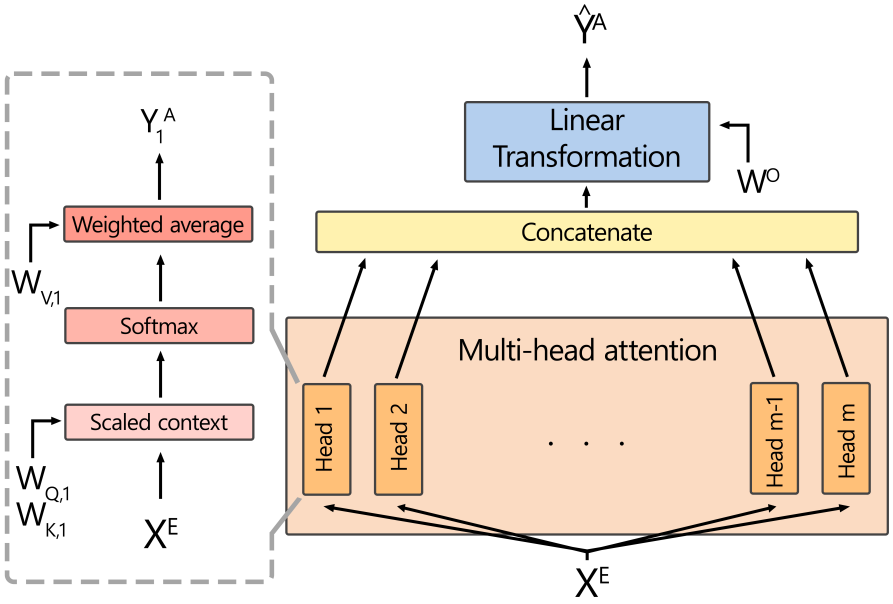


Figure 3.1: Illustration of a multi-head attention layer as a flow chart. The input X^E is projected to the m self-attention heads for multi-head attention (orange). Each of the m head follows the self-attention principle (grey) where the weight matrices used are pointed to the related block. The output from each head is concatenated (yellow) and linearly transformed (blue) to produce the resulting output $\hat{Y}^A$ from the multi-head attention layer.

3.4 Encoder

The encoder of a transformer is built as a series of encoder blocks. Each encoder block starts with a multi-head attention layer using a predefined number of heads m followed by an Add&Normalize (A&N) layer. The A&N sums the input embeddings X^E with the output from the multi-head attention layer $\hat{Y}^A$ and normalizes the sum of the embedding matrices, see Equation 3.5 [30],

$$\begin{aligned} \bar{Y}^A &= \hat{Y}^A + X^E, \\ h_{norm}(\bar{Y}^A) &= \frac{\bar{Y}^A - \mu^l}{\sqrt{\sigma^{2l}}} \end{aligned} \tag{3.5}$$

where $\bar{Y}^A$ is the intermediate residual connection followed by μ^l and σ^{2_l} which are the mean- and variance of the summed matrices respectively. Following the A&N layer is a position-wise feed-forward layer (FFN) which is a fully connected neural network applied to each position. The FFN in the network consists of two linear transformations with a rectified linear unite (ReLU) activation function connecting them, see Equation 3.6,

$$h_{FFN}(\bar{Y}^A) = \max(0, (W_1\bar{Y}^A + b_1))W_2 + b_2 \tag{3.6}$$

where W_1 and W_2 are weight matrices and b_1 and b_2 are bias vectors. Conclusively, another A&N layer is added after the FFN layer completing an encoder block. Importantly, the final output of the encoder block $\bar{Y}^A$ shares the dimensional properties with the original input X^E meaning that the output from one encoder block can be fed into another encoder block. By stacking encoder blocks, the encoder gets progressively more complex by the number of trainable parameters available to learn the task at hand. Results have shown that increasing the model complexity has led to better performance on both large tasks such as language modelling and machine translation but can also improve results on smaller scaled tasks if trained properly [30].

3.5 Pre-training with MLM

Pre-training a transformer model using the MLM technique gives the model the ability to tune trainable parameters to general patterns in unlabeled data before future fine-tuning to perform a specific task requiring data-based guidance, such as classification. Given a data set B with N_S sequences, defined as $B = \sum_{i=1}^{N_S} X_i$, a subset of the tokens in each sequence X_i will be probabilistically changed with probability p_m. The tokens have a probability of being changed to a $MASK$-token or a randomly selected token from the vocabulary [30].

For a given $X_i \in B$, the masked tokens which were randomly selected are denoted as the subset n_u with one-hot encoding vectors $x_i^u, i = 1, ..., n_u$. The MLM pretraining is done by minimizing the cross-entropy loss between selected columns of the output from the last encoder block $\bar{Y}^A$ and the one hot encoding vectors x_i^u. The selected columns from $\bar{Y}^A$ is based on which tokens are masked in the subset n_u, see Equation 3.7,

$$\arg\min_{\phi} \sum_{i=1}^{n_u} -\log(\frac{e^{\bar{y}_i^A}}{\sum_N e^{\bar{y}_i^A}})x_i^u \tag{3.7}$$

where $\bar{y}_i^A$ is the i:th column of $\bar{Y}^A$ and N is the size of the vocabulary. ϕ represents all trainable parameters in the model [2].

3.6 Transformer-based binary classification

Using the output from a transformer-based model for classification tasks often requires additional network structures added to the model. With this, the transformer

and the classification networks are trained in unity to solve a desired task. The combined model is then trained using supervised learning on labelled data with the aim of being able to accurately assign a binary label to an input sequence given to the transformer. Commonly, the classification networks are only given the *CLS*-token, or more specifically the first column of the output matrix $\bar{Y}^A$, from the encoder to base the classification decision on. This is due to computational efficacy as the dimension of the input to the classification networks without this limitation would be overwhelming depending on the size of the embedding- and sequence length. The *CLS* token is therefore trained to be an aggregated representation of the contextual information needed from the sequence for classification [2].

Training the combined model aims to minimize the binary cross-entropy loss between the predicted class x_{pred} from the classification networks and the true class x_{true}, formulated in Equation 3.8,

$$\arg\min_{\Phi} \sum_{i=1}^{N_P} x_{true,i}\log(p(x_{pred,i})) + (1 - x_{true,i})\log(1 - p(x_{pred,i}) \tag{3.8}$$

where $p(x_{pred,i})$ is the probability of the model assigning a positive class on the i:th prediction and N_P is the total amount of predictions made by the model. Φ represents all trainable parameters in the combined model [2, 19].

3.7 Overfitting & solutions

A problem which is often encountered within the field of ML is that the model learns specific patterns in the training data too well which leads to the model not being able to perform on unseen data, so called overfitting [32]. A multitude of solutions has been presented to combat overfitting, one such being the dropout technique. Dropout refers to temporarily removing units, alongside the ingoing- and outgoing signals from said units, from the network during training. This selection is done randomly with a predetermined probability p_d [33].

3.8 Principal component analysis

Principal component analysis (PCA) is a linear dimensionality reduction technique that aims to describe variance in a system of inter-correlated dependent variables. Given a data set B with p features of n individuals, a set of $p \times n$ feature vectors $b_i \in B$ can be defined. The feature vectors exist in a p-dimensional feature space and the PCA method can be interpreted as a method to rotate the coordinate axis in feature space such that the primary component is aligned with the maximum possible data dispersion. Continuing, the secondary PC-component aims to be aligned with second most the data dispersion, and so on until all p PCA components have been aligned. With this, the data can be analysed in a lower dimensional setting by studying a subset of the PC-components describing the important variance in the system. The percentage of variance each PC-component describes can be visualized in a Scree-plot. [34, 35].

3.9 Evaluation metrics

A common approach to evaluating a model trained with supervised learning is to measure the accuracy of the predictions. This entails the proportion of correctly assigned labels to the total number of predictions made by the model. However, this approach is not always the best fit to estimate how well the model is performing. Sensitivity and specificity are two metrics describing accuracy in detecting the presence or absence of a given characteristic. Both metrics are calculated by counting instances of predictions deemed *true* when the prediction is accurate to the actual outcome or *false* when the prediction is the opposite of the actual outcome. With a binary set of classes labelled *positive* and *negative*, the metric sensitivity measures the ability to correctly capture the *positive* class and specificity the *negative* class. The metrics are defined in Equation 3.9a and Equation 3.9b [36].

$$\text{Sensitivity} = \frac{\text{True positive}}{\text{True positive} + \text{False negative}} \tag{3.9a}$$

$$\text{Specificity} = \frac{\text{True negative}}{\text{True negative} + \text{False positive}} \tag{3.9b}$$

4
Methods

The following chapter will present the methodology regarding data processing, model architecture and designs of experiments for the thesis.

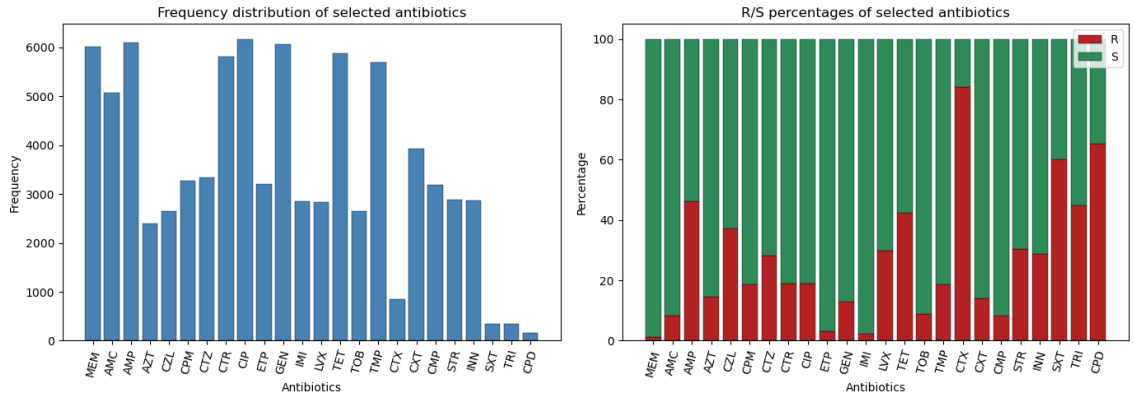
4.1 Data selection & pre-processing

Data were provided by the National Center for Biotechnology Information (NCBI). The data set was published on the 31:st of March and contained 370,673 samples of information and analysis results from genome sequencing of *Escherichia coli*. Of those, 364,820 (98.42%) of samples contained core genotype information used for pre-training and 7,701 (2.12 %) of samples contained both core genotype- and antibiotic phenotype information used for fine-tuning.

Complementing the genotype- and phenotype information, the recorded location and year of collection were added to the data set. During pre-processing, all samples missing core genotype information were removed from the data set. Moreover, an inclusion threshold was set at the year 1970 as previously recorded data were deemed unreliably collected for the study. If a gene were recorded in an incomplete way, the specific gene in question was removed. With this, the data set resulted in 364 462 samples with core genotype information and 6,486 samples with both core genotype- and phenotype information.

For the phenotype information, only recorded instances of resistant (R) or susceptible (S) for antibiotics were kept in the data set. Furthermore, to avoid creating a heavily unbalanced dataset, each antibiotic was analyzed separately where the respective frequency dictated an accepted percentage threshold of recorded instances of R and S. Three different groups were formed. Group 1 contained all antibiotics which were observed in the data set less than or equal to 100 times. Group 2 contained all antibiotics which were observed between 100- and 1000 times in the data set. Group 3 contained all antibiotics observed more than 1000 times in the data set. The threshold for the percentages was set to be 65/35 for Group 2 and 92/8 for Group 3, no matter which of the R- or S labels were in the majority. All antibiotics in the respective groups which complied with the given thresholds were kept, the rest were removed from the data set. Moreover, a threshold was not set for Group 1 as the antibiotics in this group were deemed too uncommon and were subsequently removed. Importantly, three antibiotics within the class *Carbapenems* were kept in the data set even though they did not meet the required thresholds. With this, the

frequency distribution- and R/S percentage for each of the 24 included antibiotics in the study are presented in Figure 4.1a- and Figure 4.1b respectively.



(a) Frequency distribution. **(b)** R/S percentages.

Figure 4.1: Frequency distribution- and R/S percentages for each studied antibiotic. The order of the antibiotics is consistent between subfigures.

4.2 Vocabulary

A vocabulary was built using all possible tokens that could be fed to the encoder. This includes every unique instance of different genes, locations and years recorded in the data set. In addition to this, four special tokens were included in the vocabulary. These include *CLS*, *PAD*, *MASK* and *UNK*, and were added manually as the first elements of the vocabulary. The vocabulary used in the project contained a total of 1231 tokens.

4.3 Masking sequences

Using the pre-processed data, each recorded sample was transformed into a data sequence. Each sequence starts with a *CLS*-token, followed by the year- and location for the observation and thereafter the ARG-tokens. To homogenize the sequence lengths in the data set, each sequence was extended to a predetermined length. This was set to a length of 51 to match the longest recorded sequence. Each sequence that did not reach the maximum length were extended by adding *PAD*-tokens at the end of the sequence. Moreover, if a feature of data were missing for an observation, a *PAD*-token were added in place of the missing information. These *PAD*-tokens were used as information fillers to maintain the consistency between sequence. They did not influence the training as models were passed information to only focus the self-attention on all other non- *PAD*-tokens in the sequence.

The masking process randomly selects a subset of the tokens in each sequence according to a predetermined masking probability, with a condition that at least one token in each sequence is selected. Moreover, no tokens of type *PAD* were selected.

Each selected token has an 80% chance to be replaced by a *MASK*-token, a 10% chance to be replaced by a randomly selected token from the created vocabulary and a 10% chance to remain the same. The indexes- and original tokens were saved to use for training purposes. A visualisation of the masked sequences process can be found in Figure 4.2

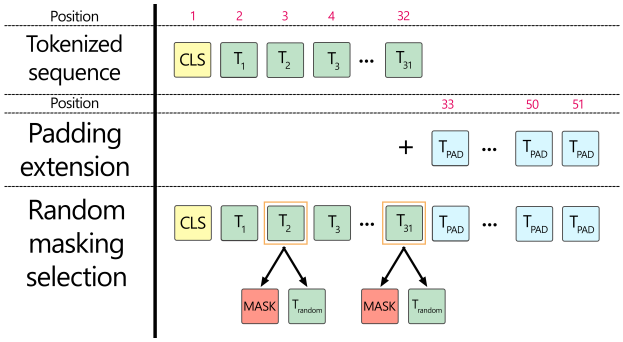


Figure 4.2: Illustration of the masked sequencing process. The three steps of the process are listed to the left. The tokens of the sequence are illustrated as labelled boxes with additional type-specific colours: *CLS*-token (yellow), ARG-token (green), *PAD*-token (blue) and *MASK*-token (red). Additional information on the position of the tokens is presented in red numbers above the respective steps of the process.

4.4 Building the model

The model framework for the thesis consisted of two parts: an encoder and a series of parallel linear classification networks, presented in Figure 4.3. The following chapters will describe the two parts in depth.

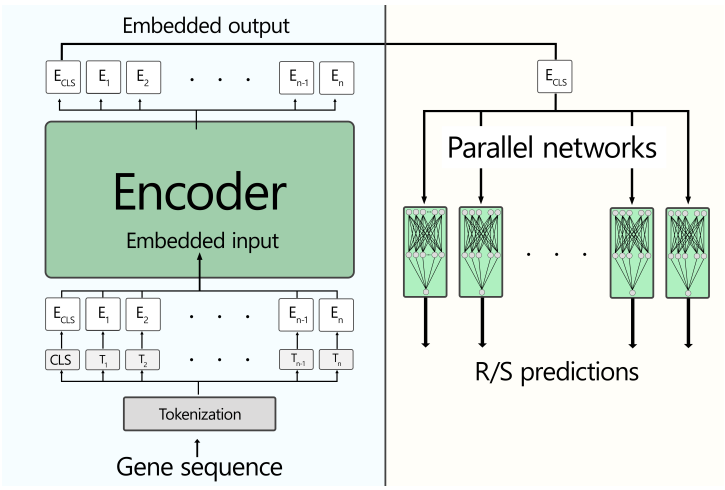


Figure 4.3: Illustration of the model presented as a flow chart. The model is divided into two parts: encoder (blue side) and parallel classification networks (yellow side). The tokenized gene sequence is shown as boxes labeled T_i and embeddings are shown as boxes labeled E_i. The embedded output of the *CLS*-token is fed to the parallel networks resulting in separate R/-predictions.

4.4.1 Encoder

The encoder was built with a set of encoder blocks, described in chapter 3.4, where the multi-head attention layer consisted of 4 attention heads and had a dropout probability p_d of 20% for all encoder blocks. Each feed-forward layer in the encoder blocks used the dimension for the embedding as the dimension for the fully connected layers. An additional linear layer was implemented after the last encoder block with the number of output neurons depending on the vocabulary size. This layer made the ARG-token predictions used during pre-training.

4.4.2 Parallel classification networks

A feed forward network was created for each unique type of antibiotic, 24 in total. These networks were placed in parallel to one another to receive the same embedded *CLS*-token output from the encoder to base the binary classification decision on. Each network was built as a fully connected layer with the same dimension as the token embeddings in the system, followed by a linear layer to produce a single output as the prediction. Moreover, a ReLU activation function and a layer normalization were implemented between the two layers.

4.5 Training

Two types of training were performed during the project. Self-supervised pre-training was performed with the MLM technique from data which contained core genotype information. Any data which contained both core genotype- and antibiotic phenotype information were excluded from the pre-training. As the pre-training only aimed to train the encoder to understand the genomic data, the parameters in the parallel networks were not tuned. Moreover, when fine-tuning the model, the models were subjected to supervised learning which used data that contained labeled genomic- and antibiotic phenotype information. Any masking done during fine-tuning aimed to increase the difficulty for the models to predict the binary labels, as the models were not trained on predictions of the missing ARG-tokens in fine-tuning.

When training the models, both during pre-training and fine-tuning, a set of hyper-parameters were fixed. To begin with, the data set was separated into two groups for training- and validation with a split percentage of 80/20 for the respective set. Each model was trained a maximum of 100 epochs with a learning rate of 10^{-7} and a batch size of 32. The loss was calculated with different functions depending on whether the aim was to pre-train or fine-tune. For pre-training, the function cross entropy loss was used and for fine-tuning a modified version of the binary cross entropy loss was used. The network was optimized using a modified version of the adaptive moment estimation (Adam) optimization algorithm with implemented weight decay. The weight decay was selected to be 10%.

All training was done on the Alvis cluster by NAISS. GPUs used in the thesis was

NVIDIA Tesla V100 for pre-training and NVIDIA A40 for fine-tuning.

4.5.1 Early stopping

A threshold was implemented to stop the training if early signs of overfitting in the network were detected. Training which continues beyond this detected point would not benefit the model which motivated the decision to stop the training to save time and computational resources. This was done by comparing calculated losses in the validation set between epochs. If the validation loss had not seen a decrease over 7 epochs the training would terminate and the model for which the lowest validation loss would be saved.

4.5.2 Selective freezing

Since most isolates in the data set only contained information regarding the susceptibility of a subset of the 24 studied antibiotics, a selective freezing method was implemented in training. For a given batch of data in the training, before using the calculated loss to update the parameters in the network the indices of which antibiotics were present in the batch was checked. This information was subsequently used to freeze the parameters in the parallel networks associated with all antibiotics not present in the respective batch to avoid being trained on information not related to them. The selective freezing process was iterative, as all parallel networks were unfrozen after the optimizer had updated the parameters for a given batch.

4.6 PCA analysis of the CLS-tokens

Fine-tuned models were further analysed by isolating the *CLS*-tokens for each isolate in the data set. The *CLS*-tokens were isolated by loading a state dict of a trained model and saving each *CLS*-token from the output of the encoder. In addition, supplementary isolate specific information such as number of genes in the sequence, presence- or absence of a specific gene and more were concatenated with the data frame of the *CLS*-tokens for future analysis.

A principle component analysis was then performed on the columns of the data frame related to the *CLS*-tokens. By selecting a subset of the calculated components which described a considerable part of the dispersion, the *CLS*-tokens could be visualized and studied as a scatter plot in two dimensions. With the supplementary information mentioned earlier, colour systems were implemented in the scatter plots to understand how the model clusters- and organizes *CLS*-tokens based on studied features.

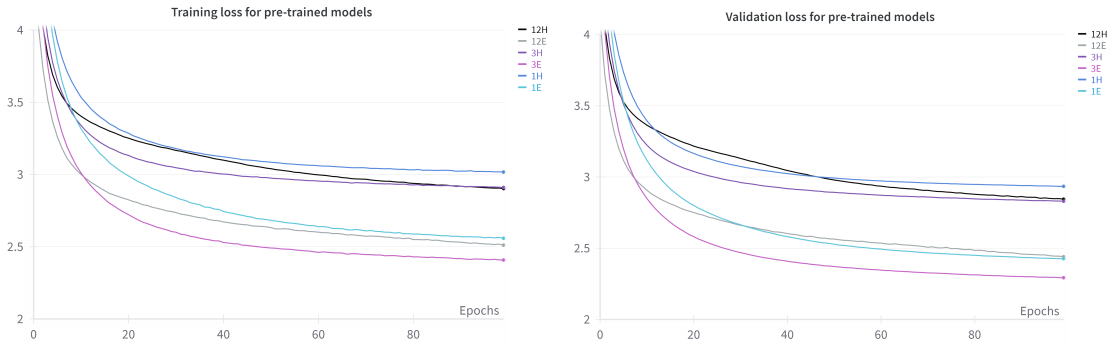
5

Results & Discussion

The following chapter will present- and discuss the gathered results from the thesis.

5.1 Pre-training

Six models with varying degree of model complexity- and masking difficulty were pre-trained according to the methodology presented in Chapter 4.5. The three complexity levels selected for pre-training were 1 encoder block, 3 encoder blocks and 12 encoder blocks. Each complexity level was trained with a 30% or 60% random masking percentage, denoted as easy- or hard training. All models used an embedding size of 512. None of the models encountered the early stopping condition and were trained for the maximum length of 100 epochs. An overview of the training- and validation losses for all pre-trained models are presented in Figure 5.1.



(a) Training loss. **(b)** Validation loss.

Figure 5.1: Training- and validation loss during pre-training for each model. The names of the models are constructed with the complexity of the model is denoted with a number (1 (blue), 3 (pink) or 12 (black)) followed by a masking difficulty denoted with a letter (E for easy, H for hard). Labels are consistent between sub-figures.

A descriptive overview of the results from the pre-training is presented in Table 5.1. The presented accuracy describes the percentage of correctly assigned ARG-tokens to the masked tokens in the sequences for the last epoch of training.

Table 5.1: Overview of the pre-training including model name, parameters, training time and resulting accuracy for each model. The models are named with a number corresponding to the encoder blocks followed by a letter indicating if the training was done with an easy (E)- or hard (H) masking percentage.

Model name	Parameters	Training time (hours:minutes)	Accuracy
1E	10,455,247	04:50	0.3942
1H	10,455,247	04:56	0.2688
3E	28,840,143	11:32	0.399
3H	28,840,143	11:37	0.2719
12E	111,572,175	17:20	0.348
12H	111,572,175	17:20	0.2593

Observing Table 5.1, the resulting accuracy of the models trained with an easy masking percentage was higher than the models trained with a hard masking percentage. This falls in line with the background, as the models with access to more information should achieve a higher performance for a given task. Notably, an increase in accuracy can be seen with an increase in model complexity between 1- and 3 encoder blocks for both masking percentages. With a further increase in number of parameters in the network, one would assume that an increase in accuracy would be observed between 3- and 12 encoder blocks as well. However, the models with 12 encoder blocks instead saw a decrease in accuracy for both masking percentages. One possible explanation for this observed phenomenon is the shared training length- and learning rate between models. If the training were adjusted to an unlimited number of epochs with the early stopping criteria, an accuracy difference dependent on complexity could possibly be found. This is supported by Figure 5.1b where the validation loss curves for models with 1- and 3 encoder blocks have stagnated more than the validation loss curves with 12 encoder blocks. This indicates that more epochs would be needed for models with 12 encoder blocks. However, this was not tested during the thesis due to constraints in computational resource management and time.

5.2 Hyperparameters in fine-tuning

Models without pre-training were fine-tuned with different hyperparameters to study the influence on model performance. The three hyperparameters tested in the thesis were embedding size, number of encoder blocks and the percentage of masking used in fine-tuning. Two experimental setups were conducted where two of the selected hyperparameters were varied and the other hyperparameter were fixed. The presented accuracy in the experiments describes the percentage of correctly assigned R/S labels for all tested antibiotics in the last epoch of training.

5.2.1 Experimental setup 1

The first experimental setup tested how the model complexity was influenced by masking percentage in fine-tuning. The models were built with varying numbers

of encoder blocks ranging from 1 block to 12 blocks with intermediate levels. All models were tested at three levels of masking percentage: 25% (easy), 50% (medium) and 75% (hard). The embedding size for all models tested in the experiment was set to 256. The recorded accuracies for all tested models in experiment 1 can be found in Figure 5.2.

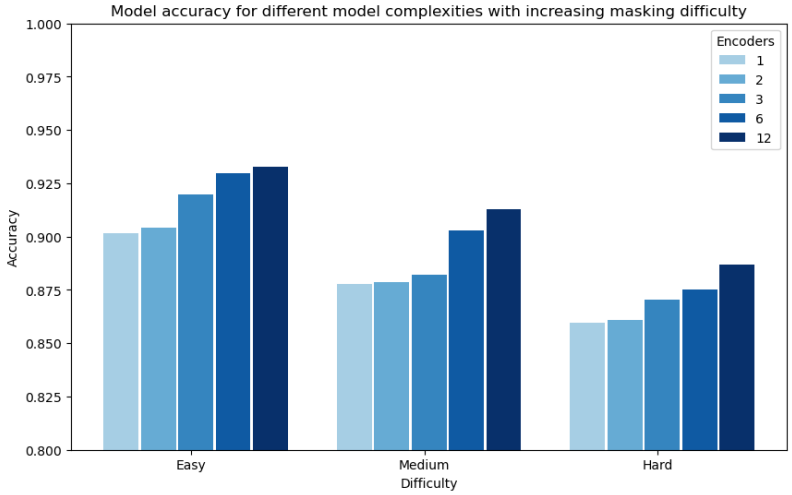


Figure 5.2: Bar diagram for the recorded accuracies for each model tested in experiment 1. The colour saturation for the bars indicates the complexity of the respective model for each masking percentage, labelled to the right.

Observing Figure 5.2, a trend between increasing model complexity and increased accuracy can be seen for all tested levels of masking percentage. The model performs better when using a lower masking percentage in fine-tuning which falls in line with the theoretical aspect of access to more information resulting in higher performance. Notably Figure 5.2 shows the importance of model complexity when limiting the information that is given to the models. A model with 12 encoder blocks that was trained on a hard masking percentage performs roughly on par with a model that has 3 encoder blocks but was trained with a medium masking percentage.

5.2.2 Experimental setup 2

The second experimental setup studied how varying embedding sizes influenced the performance for a range of model complexities. The tested embedding sizes were selected from the power of two series, ranging from 32 up to 1024. All embedding sizes were tested on three different model complexities selected to be 1, 3 and 6 encoder blocks. A fixed masking percentage of 60% was used for all models. The recorded accuracies for all tested models in experiment 2 can be found in Figure 5.3.

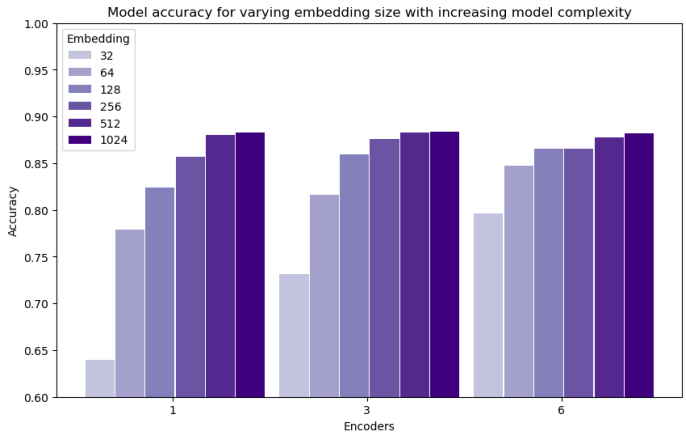


Figure 5.3: Bar diagram for the recorded accuracies for each model tested in experiment 2. The colour saturation of the bars indicates the embedding size used for each model complexity, labelled to the left.

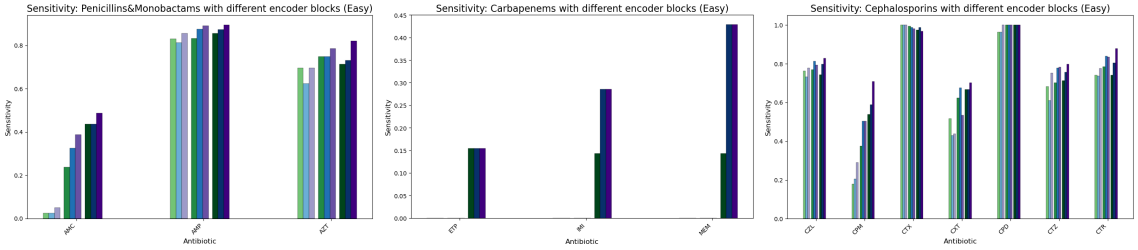
Observing Figure 5.3, a trend between increasing embedding size and increased accuracy can be seen. However, unlike in experiment 1, a stagnation in accuracy can be seen between embedding sizes 512- and 1024 for all complexity levels. This observation motivates the decision to continue using an embedding size of 512 in future experimental setups as no significant decrease in accuracy can be expected compared to selecting a larger embedding size.

5.3 Evaluating performance

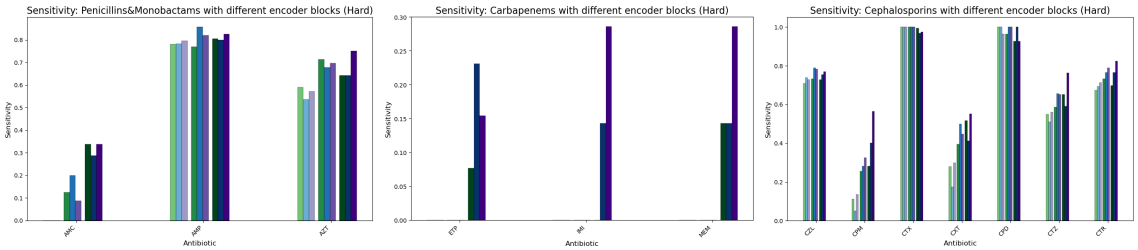
The performance seen from the models in the experimental setups presented in chapter 5.2 indicates that the models are capable of learning- and understanding genetic data. The models can then utilize that gained knowledge to assign susceptibility labels with a high accuracy depending on the selected hyperparameters. However, due to imbalances in the training data seen in Figure 4.1, a model has the possibility to score a high overall accuracy by learning to always guess the label which corresponds to the higher fraction for each antibiotic. This motivates the decision to introduce a secondary evolution metric to study model performance.

A third experiment was conducted to study differences in sensitivity- and specificity for each individual antibiotic with different model complexities. Three levels of complexity were set to 1, 3 or 12 encoder blocks. In addition to this, each complexity level was tested with two different masking percentages set as 30% (easy) or 60% (hard). Moreover, the influence of pre-training was also studied by investigating differences in sensitivity and specificity between fine-tuned models who received easy- or hard pre-training and models who were fine-tuned without pre-training. In total, the sensitivity- and specificity for all tested antibiotics for the 18 different models can be found in Table A.1-, A.2-, A.3- and A.4 presented in Appendix 1. The results from the mentioned tables were grouped based on antibiotic classes belonged to and plotted as bar diagrams. The sensitivity bar diagrams for each class are

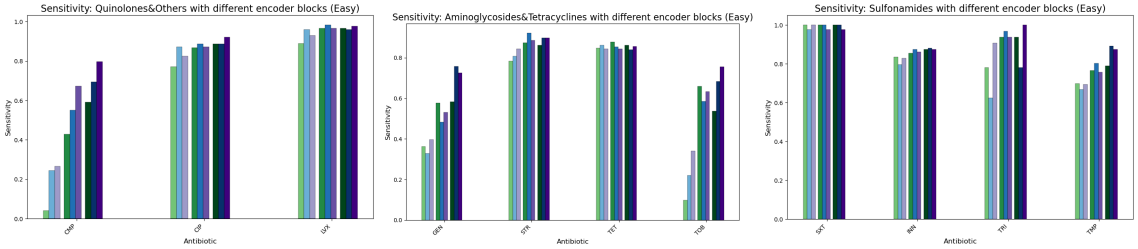
presented in Figure 5.5 and the specificity bar diagrams for each class are presented in Figure A.1 in Appendix 1.



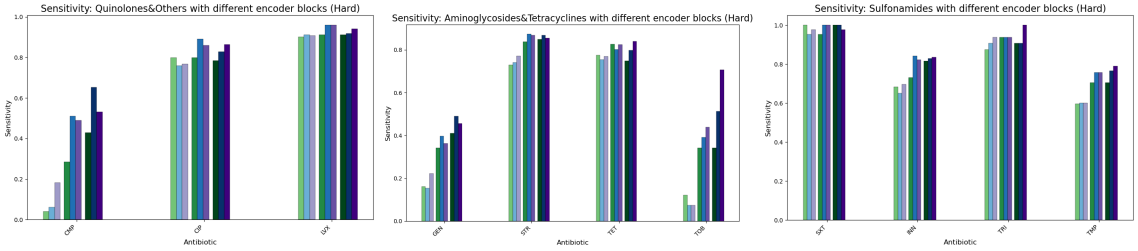
(a) Penic.&Monob., Easy. **(b)** Carbapenems, Easy **(c)** Cephalosporins, Easy.



(d) Penic.&Monob., Hard. **(e)** Carbapenems, Hard **(f)** Cephalosporins, Hard.



(g) Quino.&Others, Easy. **(h)** Amin.&Tetra., Easy . **(i)** Sulfonamides, Easy.



(j) Quino.&Others, Easy. **(k)** Amin.&Tetra., Hard . **(l)** Sulfonamides, Hard.

Figure 5.5: Barplots with sensitivity data, grouped based on antibiotic classes. Each bar is coloured based on the type of run that was performed: No pre-training (green), Easy pre-training (blue) and Hard pre-training (purple). The saturation of colours indicates the complexity of the model: 1 encoder block (light), 3 encoder blocks (medium) and 12 encoder blocks (dark). Each subplot is labelled with which type of antibiotic class is present and whether the fine-tuning was done with an easy- or hard masking difficulty.

Observing Figure 5.5, it can be concluded that some of the tested antibiotics are

easy for the model to correctly label as resistant. Such cases are LVX in panels g- and j), CPD in panels c- and f) and SXT in panels i- and l) where neither model complexity, pre-training nor fine-tuning masking percentage affected the ability to correctly predict resistances. Importantly, the high sensitivity did not limit the specificity of the antibiotics in question seen in Figure A.1 in Appendix 1. However, certain antibiotics of the tested set show clear signs of how the model complexity influences the ability to correctly predict resistance. Examples of this are in Figure 5.5h and 5.5k where all tested antibiotics show a significant increase in sensitivity for a minor trade in specificity, see Figure A.1h and Figure A.1k in Appendix 1, for both masking percentages. This observation is most notable for TOB- and GEN, which are supposedly more difficult for the model to predict resistance for compared to TET and STR. In addition to this, the effect of pre-training can also be seen in this antibiotic class as pre-trained models levels- or outperforms non pre-trained models in sensitivity for a low trade-off in specificity. This trend is evident for more complex models on the difficult antibiotics TOB- and GEN.

Continuing, certain antibiotics in the study are dependent on the model complexity to find resistances. The antibiotic AMC shows an almost 100% specificity for all models in both levels of masking percentage in Figure A.1a and Figure A.1d in Appendix 1. In contrast to this, the sensitivity for AMC seen in Figures 5.5a and 5.5d are low compared to AMP and AZT in the same group. A dependency on the model complexity can be seen for both levels of masking percentage for AMC, in particular for the hard masking percentage as only models with 3- and 12 encoder blocks have a sensitivity above 0. This indicates that models built with 1 encoder block, no matter the tested pre-training, are not complex enough to detect resistances for this antibiotic in a setting where less information is given. Continuing, a similar conclusion can be drawn for the antibiotics in class *carbapenems*. Observing Figure 5.5b and 5.5e, only models with 12 encoder blocks are able to correctly find resistances to antibiotics ETP, IMI and MEM which are the antibiotics with the most imbalanced R/S percentages in the data set, see Figure 4.1. This answers an important question regarding the importance of model complexity in the field: Only very complex models have the capacity to accurately predict resistance from antibiotics where the occurrence of resistance is rare. Moreover, the importance of pre-training is also displayed in the same group as pre-training resulted in up to 3 times the sensitivity without significant loss in specificity, see Figure A.1b and Figure A.1e in Appendix 1.

Observing Figure 5.5, the use of pre-training has an overall positive effect on the sensitivity for a number of antibiotics as seen in the cases presented above. However, drawing a generalized conclusion on which type of pre-training is better for the system is difficult. Observing CMP in Figure 5.5g and 5.5j, an increase in sensitivity can be seen for all model complexities at both masking percentages with pre-training without a significant loss of specificity, see Figure A.1g and Figure 5.5j in Appendix 1. However, the type of pre-training which resulted in the highest sensitivity differs. For fine-tuning with an easy masking percentage, the models benefited from training on a hard masking percentage and the opposite was found when studying the fine-tuned models on a hard masking percentage. This pattern is not found

when analyzing other antibiotics in the study. Examples of this are antibiotics in the class *cephalosporins* in Figure 5.5c. For both CXT and CTR, the models with 3 encoder blocks benefited more from an easy masking percentage during pre-training compared to the models with 12 encoder blocks which benefited more from a harder masking percentage during pre-training. Even though the best type of pre-training was situational, a common denominator between all three examples presented is that both types of pre-training outperformed no pre-training at all which underlines the importance.

5.4 Analyzing learning patterns

A PCA-CLS analysis was performed on all models built with 12 encoder blocks from Chapter 5.3 to understand important features of the data that influence the decisions made by the models. The number of PC-components to analyze was decided based on the complimentary Scree-plots produced for each PCA-CLS analysis, see Figure A.2 in Appendix 1. Based on Figure A.2 in Appendix 1, the first 3 PC-components were selected to be studied further. The 100 data points with the highest- and lowest values for each PC-component were selected as points of interest, denoted the *Top*- and *Bot* group respectively. These points are of interest as they have significant disparity in the direction of the studied component. All chromosomal genes with point mutations are presented as the original amino acid, the position of the mutation followed by the substituted amino acid.

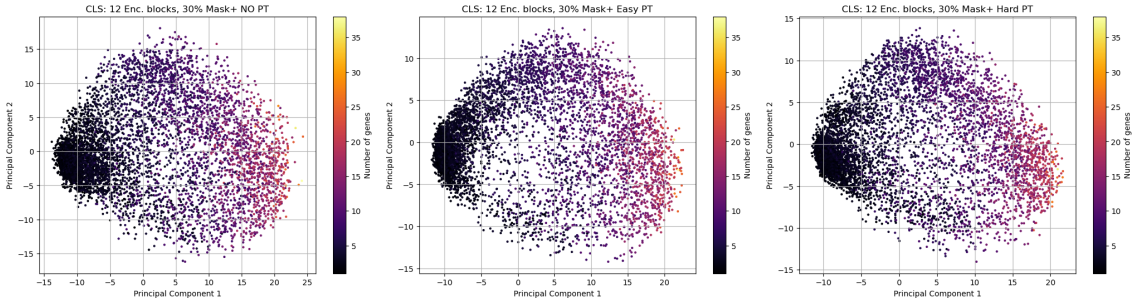
5.4.1 PC1: ARG quantity

The ARG content for the points of interest for PC-component 1 (PC1) was analyzed by counting the number of occurrences for each ARG in the *Top*- and *Bot* groups. With this, a list of the top 5 most common ARGs in each group was created, presented in Table 5.2.

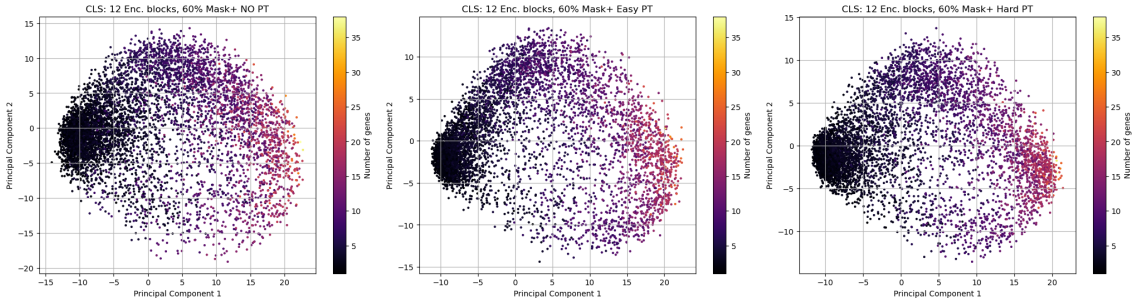
Table 5.2: The five most common ARGs in the *Top*- and *Bot* groups of points of interest for all tested models in PC1. The number of occurrences for each ARG is presented in parenthesis after the ARG.

Top	1	2	3	4	5
12E-N	glpT-E448K(92)	gyrA-S83L(89)	parC-S80I(85)	gyrA-D87N(82)	aph(3")-Ib(72)
12E-E	parC-S80I(97)	gyrA-S83L(97)	gyrA-D87N(92)	glpT-E448K(88)	aph(3")-Ib(81)
12E-H	gyrA-S83L(97)	parC-S80I(96)	gyrA-D87N(93)	glpT-E448K(91)	aph(3")-Ib(85)
12H-N	glpT-E448K(93)	parC-S80I(90)	gyrA-S83L(90)	gyrA-D87N(88)	sul1(71)
12H-E	gyrA-S83L(97)	parC-S80I(96)	glpT-E448K(95)	gyrA-D87N(95)	sul1(87)
12H-H	parC-S80I(99)	gyrA-S83L(99)	gyrA-D87N(96)	glpT-E448K(91)	sul1(85)
Bot	1	2	3	4	5
12E-N	glpT-E448K(96)	pmrB-Y358N(30)	pmrB-E123D(9)	cyaA-S352T(8)	marR-S3N(8)
12E-E	glpT-E448K(99)	pmrB-Y358N(20)	pmrB-E123D(18)	marR-S3N(16)	cyaA-S352T(12)
12E-H	glpT-E448K(98)	pmrB-Y358N(52)	marR-S3N(26)	pmrB-E123D(21)	uhpT-E350Q(2)
12H-N	glpT-E448K(91)	pmrB-Y358N(23)	marR-S3N(22)	pmrB-E123D(15)	cyaA-S352T(4)
12H-E	glpT-E448K(99)	pmrB-Y358N(27)	pmrB-E123D(26)	marR-S3N(11)	cyaA-S352T(8)
12H-H	glpT-E448K(99)	pmrB-Y358N(22)	marR-S3N(16)	pmrB-E123D(12)	cyaA-S352T(8)

Observing Table 5.2, the common genetic content differs between groups. ARGs such as point mutations in the *gyrA*-gene and the point mutation *parC-S80L* are common occurrences for the *Top*-group for all models. However, the quantity of common ARGs in the *Bot* group is lower. Out of the 100 points of interest, only one out of six models has counted the 5th most common ARG 10 times or more compared to the *Top*-group where all models counted the 5th most common ARG more than 70 times. This indicates that the overall ARG quantity in the *Bot*-group is generally lower. Moreover the ARG that stands out for all models in the *Bot*-group, the point mutation *glpT-E448K*, can also be found as a common ARG in the *Top*-group which means that the ARG itself is a common occurrence and might not be PC1 dependent. With this, the *CLS*-tokens for all sequences for each model were visualized as a scatter plots in with additional colouring dependent on the number of ARGs each isolate had, see Figure 5.6.



(a) Easy FT, No PT. **(b)** Easy FT, Easy PT. **(c)** Easy FT, Hard PT.



(d) Hard FT, No PT. **(e)** Hard FT, Easy PT. **(f)** Hard FT, Hard FT.

Figure 5.6: Scatter plot of PC1 & PC2 for each *CLS* token from the tested models. Each *CLS* token for all subplots has been coloured depending on the number of ARGs for the respective sequence. Colour palettes are consistent between subfigures. Each subplot is captioned with the settings used in training to produce the model.

Observing Figure 5.6, a dependency of the quantity of ARGs along the PC1 axis can be seen for all tested models. A large group of data points with a lower ARG count

has been grouped on the negative side of the PC1-axis with an arc shape towards the positive PC1-axis where points with higher ARG counts are located for all models. As all Scree-plots in Figure A.2 indicates that PC1 explains a significant aspect of the disparity of the *CLS*-tokens, the models base a lot of the decisions on the number of present ARGs. This observation is logical in the sense that the number of ARGs in bacteria is one of the more fundamental aspects of the data. Having more ARGs in a sequence gives the models more information to base the resistance predictions on, and therefore, makes the quantity of ARGs the most important factor for the models.

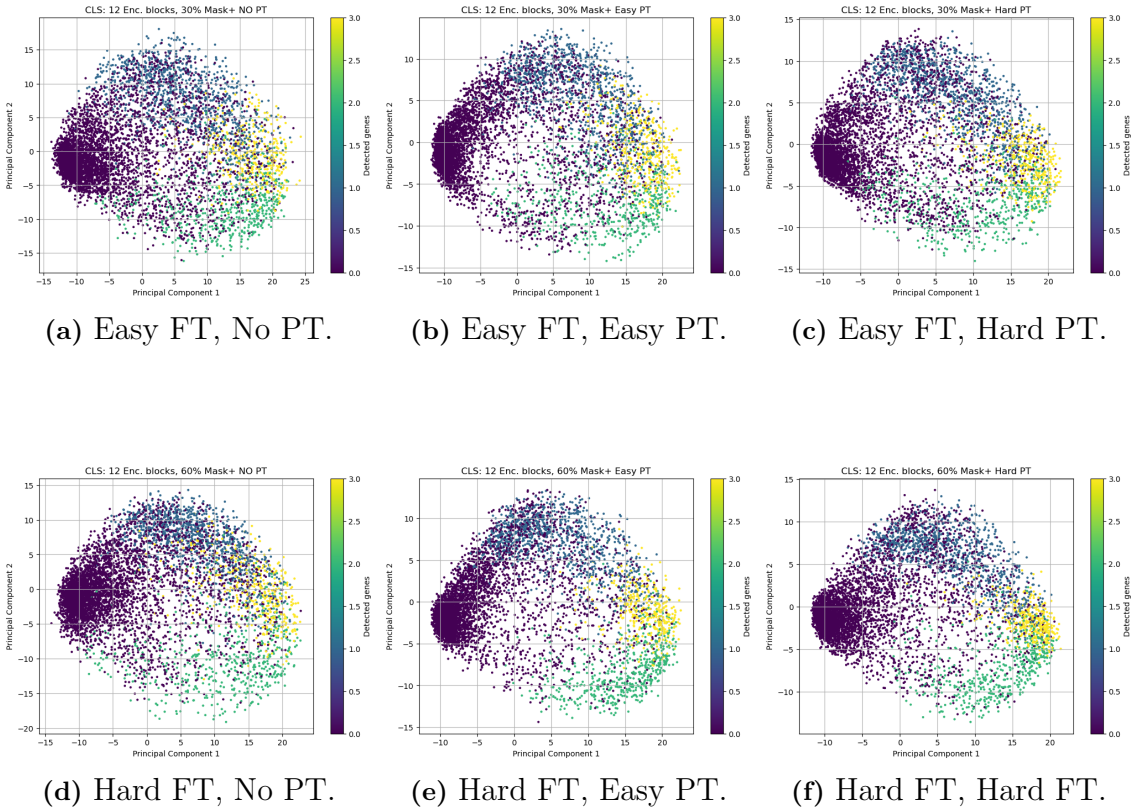
5.4.2 PC2: ARG groupings

Similar to PC1, the occurrences of ARGs were counted for the points of interest for PC-component 2 (PC2). The list of the top 5 most common ARGs in the *Top*- and *Bot* groups can be found in Table 5.3.

Table 5.3: The five most common ARGs in the *Top*- and *Bot* groups of points of interest for all tested models in PC2. The number of occurrences for each ARG is presented in parenthesis after the ARG.

Top	1	2	3	4	5
12E-N	gyrA-S83L(97)	parC-S80I(96)	gyrA-D87N(92)	glpT-E448K(85)	uhpT-E350Q(77)
12E-E	gyrA-S83L(100)	gyrA-D87N(95)	parC-S80I(95)	glpT-E448K(80)	uhpT-E350Q(71)
12E-H	gyrA-S83L(99)	gyrA-D87N(98)	parC-S80I(98)	glpT-E448K(85)	uhpT-E350Q(72)
12H-N	gyrA-D87N(100)	gyrA-S83L(100)	parC-S80I(99)	glpT-E448K(92)	uhpT-E350Q(87)
12H-E	gyrA-S83L(100)	parC-S80I(99)	gyrA-D87N(96)	uhpT-E350Q(77)	pmrB-E123D(71)
12H-H	gyrA-S83L(94)	glpT-E448K(86)	parC-S80I(85)	gyrA-D87N(81)	uhpT-E350Q(75)
Bot	1	2	3	4	5
12E-N	glpT-E448K(95)	aph(3")-Ib(69)	aph(6)-Id(67)	tet(B)(63)	sul2(61)
12E-E	glpT-E448K(94)	aph(3")-Ib(74)	aph(6)-Id(70)	sul2(67)	tet(B)(62)
12E-H	blaTEM-1(90)	glpT-E448K(89)	aph(3")-Ib(73)	aph(6)-Id(70)	tet(B)(63)
12H-N	aph(3")-Ib(92)	aph(6)-Id(90)	glpT-E448K(89)	blaTEM-1(73)	tet(B)(71)
12H-E	aph(3")-Ib(87)	blaTEM-1(85)	aph(6)-Id(84)	glpT-E448K(83)	sul2(71)
12H-H	glpT-E448K(93)	blaTEM-1(92)	aph(3")-Ib(69)	aph(6)-Id(67)	tet(A)(62)

Observing Table 5.3, a quantity difference of ARGs between group *Top* and group *Bot* is not as apparent as in the case of PC1. This indicate that PC2 is not as dependent on the number of ARGs in a sequence but rather the type of ARG present in a sequence. From Table 5.3 a trend can be observed that the two point mutations of *gyrA* and the point mutation *parC-S80I* are common occurrences in the *Top*-group with almost every isolate carrying all three ARGs. Moreover, a similar pattern can be observed in the *Bot*-group as the two *aph* ARGs are prevalent in most of the points of interest. This dependency was studied by plotting the *CLS*-tokens in PC1- and PC2 with additional colouring dependent on which ARGs each data point carried. The isolates were given separate colours depending on if they had both the *gyrA-D87N* and *parC-S80I* mutations, the *aph(3")-Ib* and *aph(6)-Id* or none of the ARGs. In addition, a fourth colour was introduced for data point which carried all four of the previously mentioned ARGs. The scatter plots for PC1&PC2 with the introduced colour system for all studied models can be found in Figure 5.7.



(a) Easy FT, No PT. **(b)** Easy FT, Easy PT. **(c)** Easy FT, Hard PT.

(d) Hard FT, No PT. **(e)** Hard FT, Easy PT. **(f)** Hard FT, Hard FT.

Figure 5.7: Scatter plot of PC1 & PC2 for each *CLS* token from the tested models. Each *CLS* token for all subplots has been coloured depending on the ARG content of the sequence. Colour palettes are consistent between subfigures. Isolates with none of the studied point mutations (purple), isolates with *gyrA-D87N* and *parC-S80I* (blue), isolates with *aph(3")-Ib* and *aph(6')-Id* (green) and isolates with all studied point mutation (yellow) are marked accordingly. Each subplot is captioned with the settings used in training to produce the model.

Observing Figure 5.7, a dependency of the tested ARG groups along the PC2 axis can be seen. All models separate the isolates containing the *gyrA-D87N* and *parC-S80I* mutations from the isolates *aph(3")-Ib* and *aph(6)-Id*. Any overlapping regions between the studied ARG groups either contain both of the groups or none of them. Point mutations in the *gyrA* and *parC* genes in *E. coli* encodes for resistance to antibiotics in the *quinolones* class. In order to gain high level resistance against the *quionolones* containing fluorine the bacteria needs more than one mutation in the genetic content involved in the maintenance of DNA topology. Two of these genes are *gyrA* and *parC* which explains why these ARGs are found together in the data [37]. On the other hand, the mutations in the *aph* genes encode for resistance to antibiotics in the class *aminoglycosides* [38]. With the significant segregation of data the models have performed, a difference in classes that the studied ARGs encode resistances for is expected. What is not yet understood is why the models weigh resistance profiles to these classes so heavily compared to the other available classes, such as *penicillins*. Moreover, other ARG which also encodes for resistance against the highlighted antibiotic classes in PC2 are not prioritized by the model.

The *qnr* ARGs also encodes for resistance to *quinolones* but can not be found in the *Top*-group where the other *quinolones* ARGs are present [39].

5.4.3 PC3: pmrB mutations

Finally, the ARG content for the points of interest for PC-component 3 (PC3) were analyzed following the same methodology as previously mentioned. A list of the top 5 most common ARGs in each group for the third PC axis can be found in Table 5.4.

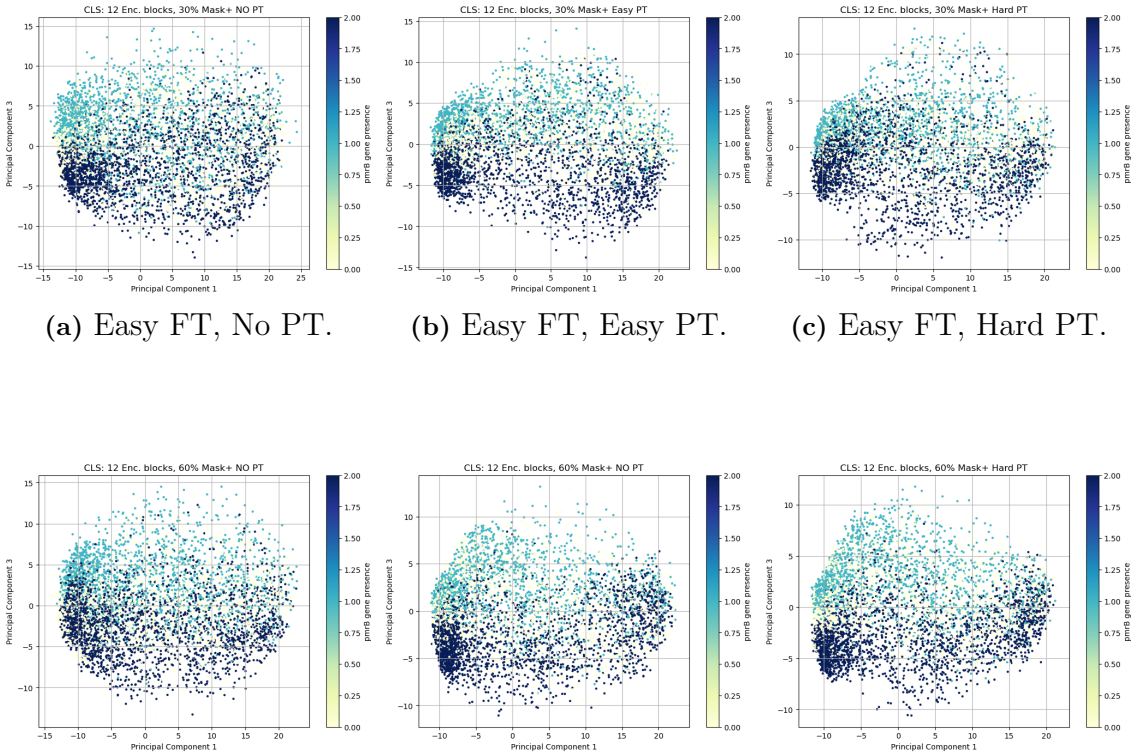
Table 5.4: The five most common ARGs in the *Top*- and *Bot* groups of points of interest for all tested models in PC3. The number of occurrences for each ARG is presented in parenthesis after the ARG.

Top	1	2	3	4	5
12E-N	glpT-E448K(98)	pmrB-Y358N(65)	blaCMY-2(54)	tet(B)(27)	sul2(25)
12E-E	glpT-E448K(94)	blaCMY-2(54)	pmrB-Y358N(47)	tet(A)(42)	cyaA-S352T(27)
12E-H	glpT-E448K(96)	blaCMY-2(81)	pmrB-Y358N(44)	cyaA-S352T(34)	uhpT-E350Q(15)
12H-N	glpT-E448K(97)	blaCMY-2(54)	pmrB-Y358N(45)	cyaA-S352T(37)	qnrS1(17)
12H-E	glpT-E448K(67)	tet(B)(40)	qnrS1(38)	sul1(37)	aadA2(34)
12H-H	glpT-E448K(90)	tet(B)(80)	pmrB-Y358N(62)	aph(6)-Id(43)	aph(3")-Ib(43)
Bot	1	2	3	4	5
12E-N	gyrA-S83L(86)	parC-S80I(75)	mph(A)(71)	sul2(71)	glpT-E448K(69)
12E-E	pmrB-E123D(85)	gyrA-S83L(84)	uhpT-E350Q(83)	blaTEM-1(74)	dfrA17(71)
12E-H	gyrA-S83L(100)	gyrA-D87N(99)	parC-S80I(99)	uhpT-E350Q(88)	pmrB-E123D(86)
12H-N	pmrB-E123D(98)	uhpT-E350Q(96)	gyrA-S83L(95)	gyrA-D87N(91)	parC-S80I(90)
12H-E	pmrB-E123D(83)	blaTEM-1(78)	glpT-E448K(76)	uhpT-E350Q(71)	gyrA-S83L(65)
12H-H	pmrB-E123D(82)	uhpT-E350Q(76)	marR-S3N(62)	blaTEM-1(58)	gyrA-S83L(55)

Observing Table 5.4, a dependency similar to what was found in PC2 between two groups of ARGs is not as apparent for PC3. The diversity of ARGs in the points of interest is higher than in PC1 and PC2, with the most consistent ARG being *glpT-E448K* in the *Top*-group similar to PC1. However, as discussed previously, the frequent nature of this ARG complicates any further analysis of it being an important factor for PC3. As can be seen in Table 5.4, *glpT-E448K* can also be found in the *Bot*-group with lower counts. The ARG grouping of *gyrA* / *parC* point mutations seen in PC2 is common as well in PC3. However, the consistency of how common the ARG group is between models is not as apparent as in PC2. Moreover, the segregation between *gyrA* / *parC* and the *aph* ARGs seen in PC2 is not present in PC3. This indicates that the *gyrA* / *parC* mutations might be influential to PC3 but not as important as previously studied in PC2.

An ARG that may be more influential to PC3 however is the point mutation *pmrB-E123D* which is prevalent in the *Bot*-group, seen in Table 5.4. This ARG is common for five out of six tested models, also being the most common ARG for all models fine-tuned with a 60% masking percentage. Interestingly, another point mutation of *pmrB* is alo common for five out of six models in the *Top*-group. The relationship between the point mutations *pmrB-E123D* and *pmrB-Y358N* was studied by plotting the *CLS*-tokens in PC1- and PC3 with additional colouring dependent on the ARGs in each isolate. One colour was given to all the isolates carrying *pmrB-E123D* and one colour was given to all isolates carrying *pmrB-Y358N*. In addition, separate

colours were also given to the isolates carrying both ARGs- or none of them. The resulting scatter plots can be found in Figure 5.8



(a) Easy FT, No PT. **(b)** Easy FT, Easy PT. **(c)** Easy FT, Hard PT.

(d) Hard FT, No PT. **(e)** Hard FT, Easy PT. **(f)** Hard FT, Hard FT.

Figure 5.8: Scatter plot of PC1 & PC3 for each *CLS* token from the tested models. Each *CLS* token for all subplots has been coloured depending on the ARG content of the sequence. Isolates with none of the studied point mutations (beige), isolates with pmrB-Y38N (aqua), isolates with pmrB-E123D (navy) and isolates with all studied point mutations (black) are marked accordingly. Colour palettes are consistent between subfigures. Each subplot is captioned with the settings used in training to produce the model.

Observing Figure 5.8, a dependency between the two point mutations of *pmrB* can be seen in PC3 for all tested models. Data points carrying *pmrB-Y38N* are mostly found on the positive PC3-axis in comparison to data points with *pmrB-E123D* located more towards the negative side. This trend in PC3 suggests that the *pmrB*-mutations are an important basis for the decisions the model makes. The segregation of data dependent on *pmrB* is not ideal as regions with overlap exist, and unlike the case of the ARG groupings in PC2, these overlapping regions do not contain a combination of the studied ARGs. In fact, Figure 5.8 shows that no isolates in the data set carry both the studied *pmrB*-mutations. The hypothesis regarding this observation is that the *pmrB* gene in *E. coli* does not benefit from the amino acid change from both mutations simultaneously whereas having one of them can be advantageous. Similar observations have been made when studying how different

numbers of amino acid changes affect the structure of proteins [40].

The functionality of the *pmrB* gene is connected to reducing negative charges in the outer monolayer of the membrane in the bacteria. Mutations in the gene have been studied in relation to resistance to Colistin, an antibiotic in the class *Polymyxins* which often is regarded as a last resort option due to the possible side effects during treatment [41]. Importantly, as can be seen in Figure 4.1, due to the rare nature of usage for Colistin the antibiotic did not meet the required thresholds to be included in the thesis. This means that the models weigh these *pmrB* point mutations for reasons which is unrelated to Colistin resistance. Reasons as to why this phenomenon appears can only be speculative as research regarding *pmrB* mutations is centered around resistance to *polymyxins* which can not be the case for this thesis as no antibiotics for this class are available. This leads to the idea that point mutations in the *pmrB* gene may play a larger role in antibiotic resistance than previously anticipated by potentially facilitating other ARGs. Relationships between *pmrB* point mutations and other ARGs could possibly be found in the observed overlapping regions in the PC3 plots in Figure 5.8 as the overlapping regions themselves indicate that PC3 is most likely dependent on something additional to the *pmrB* mutations.

5.5 Future work

Further experimentation with how to better tailor models to the problem and the data at hand was not conducted due to time constraints. Studying the potential impact of other types of loss functions, varying the number of attention heads in encoder blocks and such implementations has on even rarer antibiotics which did not meet the requirement set in the thesis could be beneficial. In general, the goal should be to introduce more antibiotics to the models from a clinical perspective to give a more well-rounded treatment plan for patients. The thesis saw an increase in sensitivity for a number of antibiotics with rare occurrences of resistance when model complexity increased. Pushing the model complexity even further with more encoder blocks could see resistance to the rare antibiotics being detected even better up to a possible threshold. Furthermore, the positive impact of pre-training was observed in the thesis. As discussed in Chapter 5.1, the models had the potential to be pre-trained even longer as the training was forcefully stopped after 100 epochs. Letting the pre-trained models train until the early stopping criteria is met may be resource demanding but could improve results and give a clearer answer to which type of pre-training is more beneficial to the system. Lastly, the thesis only studied data from the species *E. coli*. Branching out and studying genomic data from more than one species opens the possibility for the model to be able to accurately predict the type of species encountered in a clinical setting as well as the types of antibiotics the individuals are resistant towards.

The thesis highlighted some of the ARGs the models deemed as more important when predicting resistance profiles. The reason as to why the model weights these ARGs so highly compared to other ARGs in the studied data set is yet to be known.

Understanding why the relationship between *gyrA* / *parC* and the *aph* ARGs is so important or what point mutations in *pmrB* provide for resistance profiles not related to *polymyxins* may be instrumental to gain a deeper understanding of antibiotic resistance. However, this gained knowledge of what the models are trying to teach us is something that most likely needs to be further studied in a laboratory settings.

6
Conclusion

The thesis aimed to implement a transformer-based AI method that utilizes genomic data to predict resistance to commonly used antibiotics. The models that were created using this methodology managed to correctly predict the resistance profiles of bacteria upwards to 94% depending on the selection of hyperparameters- and model complexity. As the R/S percentages for all antibiotics in the data set used for the thesis were unbalanced even after introducing exclusion criteria, further analysis with different metrics highlighted the importance of a high model complexity. In particular, certain classes of antibiotics like *carbapenems* where resistance are a rare occurrence in the data set only captured by the most complex models tested. The thesis concluded that the use of 12 encoder blocks is beneficial for the model and encourages future studies regarding increasing the model complexity even further. Moreover, the thesis can also conclude that the use of self-supervised pre-training with genetic data improves the ability of a complex model to accurately find resistance profiles in bacteria. Even though models with both types of pre-training consistently performed better than models without, the type of pre-training that performed better was situational which means that no conclusion was drawn on which type should be preferred.

After studying the disparity of the *CLS*-tokens produced by trained models, learning patterns revealed important aspects of what the decisions are based on. To begin with, the most important characteristic of the data to the model is how many ARGs are present in a sequence. This fundamental aspect of the data influences most of the decisions made by the models. Moreover, the lower dimensions of the PCA highlighted how certain ARGs are more influential in the decisions the model makes than others. One such finding in the thesis was a relationship between two groups of ARGs: *gyrA / parC* encoding resistance to *flouroquinolone* and *aph(3")-Ib / aph(6)-Id* encoding resistance to *aminoglycosides*. Furthermore, the point mutations *E123D-* and *Y358N* in the gene *pmrB* also showed significant influence in the decisions made by the models. These genetic findings that are based on patterns the models have found are of interest to understanding antibiotic resistance better and are encouraged to be further studied in a laboratory setting.

Bibliography

[1] S. Dattani, L. Rodés-Guirao, H. Ritchie, E. Ortiz-Ospina, M. Roser. *Life Expectancy*. Our World in Data. 2023. Available from: https://ourworldindata.org/life-expectancy

[2] J. Inda-Díaz. *New AI-based methods for studying antibiotic-resistant bacteria*. Gothenburg: University of Gothenburg, Faculty of Science; 2023. Available from: https://hdl.handle.net/2077/78675

[3] W.A. Adedeji. *The treasure called antibiotics*. Ann Ib Postgrad Med. 2016 Dec;14(2):56-57. PMID: 28337088; PMCID: PMC5354621.

[4] *What is the total consumption of antibiotics?* [Internet]. Federal Office of Public Health (FOPH). [updated 06.12.2023]. Available from: https://www.bag.admin.ch/bag/en/home/krankheiten/infektionskrankheiten-bekaempfen/antibiotikaresistenzen/wie-viele-antibiotika-verbrauchen-wir—.html

[5] M. Lobanovska, G. Pilla. *Penicillin's Discovery and Antibiotic Resistance: Lessons for the Future?* YALE JOURNAL OF BIOLOGY AND MEDICINE . 2017 Mar 29;90(1):135-145. PMID: 28356901; PMCID: PMC5369031

[6] *Antimicrobial resistance* [Internet]. World Health Organization (WHO). [updated 21.11.2023]. Available from: https://www.who.int/news-room/fact-sheets/detail/antimicrobial-resistance

[7] M. Naghavi, *et al.*. *Global burden of bacterial antimicrobial resistance in 2019: a systematic analysis*. Lancet. 2022 Feb 12–18;397(10274):533-43.

[8] World Bank. Drug-Resistant Infections: A Threat to Our Economic Future. Washington, DC: World Bank; 2017. License: Creative Commons Attribution CC BY 3.0 IGO.

[9] G. Cox, G.D. Wright. *Intrinsic antibiotic resistance: Mechanisms, origins, challenges and solutions*. Int J Med Microbiol. 2013;303(6-7):287-292. doi:10.1016/j.ijmm.2013.02.009.

[10] E. Peterson, P. Kaur. *Antibiotic resistance mechanisms in bacteria: relationships between resistance determinants of antibiotic producers, environmental bacteria, and clinical pathogens*. Frontiers in Microbiology. 2018 Nov 30;9:2928. doi: 10.3389/fmicb.2018.02928.

[11] A. MacGowan, E. Macnaughton. *Antibiotic resistance*. Medicine. 2017;45(10):622-628. doi: 10.1016/j.mpmed.2017.07.006.

[12] M. Frieri, K. Kumar, A. Boutin. *Antibiotic resistance*. J Infect Public Health. 2017;10(4):369-378. doi: 10.1016/j.jiph.2016.08.007.

[13] I. Gajic, J. Kabic, D. Kekic, M. Jovićević, M. Milenkovic, D. Mitić-Ćulafić, A. Trudic, L. Ranin, N. Opavski. *Antimicrobial susceptibility testing: a com-*

prehensive review of currently used methods. Antibiotics. 2022 Mar;11:427. doi: 10.3390/antibiotics11040427.

[14] M.F. Anjum, E. Zankari, H. Hasman. *Molecular methods for detection of antimicrobial resistance*. Microbiol Spectrum. 2017;5(6). doi: 10.1128/microbiolspec.arba-0011-2017.

[15] C. Köser. *Whole-genome sequencing to control antimicrobial resistance*. Trends Genet. 2014;30(9)

[16] T. Davenport, R. Kalakota. *The potential for artificial intelligence in healthcare*. Future Healthcare Journal. 2019 Jun;6(2):94-98. doi: 10.7861/futurehosp.6-2-94

[17] R. Forghani, ed. *Machine Learning and Other Artificial Intelligence Applications, An Issue of Neuroimaging Clinics of North America*, E-Book. Elsevier Health Sciences; 2020. ISBN: 9780323712453.

[18] V. Kaul, S. Enslin, S.A. Gross. *History of artificial intelligence in medicine*. Gastrointestinal Endoscopy 2020;92(4):807-812. doi: 10.1016/j.gie.2020.06.040

[19] I. Goodfellow, Y. Bengio, A. Courville. *Deep Learning*. MIT Press; 2016.

[20] A. Gillioz, J. Casas, E. Mugellini, O. Abou Khaled. *Overview of the Transformer-based Models for NLP Tasks*. In: Proceedings of the Federated Conference on Computer Science and Information Systems. 2020:179-183. doi: 10.15439/2020F20.

[21] A. Vaswani, *et al.*. *Attention Is All You Need*. In: 31st Conference on Neural Information Processing Systems (NIPS 2017); Long Beach, CA, USA.

[22] J.K. Tripathy, *et al.*. *Comprehensive analysis of embeddings and pretraining in NLP*. Computer Science Review. 2021;42:100433. doi: 10.1016/j.cosrev.2021.100433.

[23] O. Ozyegen, H. Jahanshahi, M. Cevik, *et al.* Classifying multi-level product categories using dynamic masking and transformer models. J Data Inf Manag. 2022;4:71-85. doi: 10.1007/s42488-022-00066-6.

[24] R. Irwin, S. Dimitriadis, J. He, E.J. Bjerrum. *Chemformer: a pre-trained transformer for computational chemistry*. Machine Learning: Science and Technology. 2022;3(1):015022. doi: 10.1088/2632-2153/ac3ffb.

[25] S. Jamil, M. Jalil Piran, O.J. Kwon. *A Comprehensive Survey of Transformers for Computer Vision*. Drones. 2023;7(5):287. doi:10.3390/drones7050287

[26] K.M. Ruff, R.V. Pappu. *AlphaFold and implications for intrinsically disordered proteins*. JMB Journal of Molecular Biology. 2021;433(20):167208. doi: 10.1016/j.jmb.2021.167208.

[27] C. Zhang, S. Liwicki, R. Cipolla. *Image reranking using pretrained vision transformers*. Cambridge Research Lab. 2022; University of Cambridge, UK.

[28] P. Shaw, J. Uszkoreit, A. Vaswani. *Self-attention with relative position representations*. In: North American Chapter of the Association for Computational Linguistics; 2018. Available from: https://api.semanticscholar.org/CorpusID:3725815.

[29] L. Liu, J. Liu, J. Han. *Multi-head or Single-head? An Empirical Comparison for Transformer Training*. arXiv:2106.09650 [cs.CL]. 2021.

[30] J. Devlin, M.W. Chang, K. Lee, K. Toutanova. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Compu-

tational Linguistics: Human Language Technologies; 2019. p. 4171–4186. doi: 10.18653/v1/N19-1423.

[31] J.L. Ba, J.R. Kiros, G.E. Hinton. *Layer Normalization*. arXiv:1607.06450 [stat.ML]. 2016.

[32] X. Ying. *An Overview of Overfitting and its Solutions*. Journal of Physics. 2019;1168:022022. DOI: 10.1088/1742-6596/1168/2/022022.

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research. 2014;15:1929-1958.

[34] I.T. Jolliffe, J. Cadima. *Principal component analysis: a review and recent developments*. Phil. Trans. R. Soc. A. 2016;374:20150202. DOI: 10.1098/rsta.2015.0202.

[35] F.L. Gewers, G.R. Ferreira, H.F. De Arruda, F.N. Silva, C.H. Comin, D.R. Amancio, D.R. Costa. *Principal Component Analysis: A Natural Approach to Data Exploration*. ACM Comput. Surv. 2021;54(4):70:1-34. DOI: 10.1145/3447755.

[36] T.F. Monaghan, S.N. Rahman, C.W. Agudelo, A.J. Wein, J.M. Lazar, K. Everaert, R.R. Dmochowski. *Foundational Statistical Principles in Medical Research: Sensitivity, Specificity, Positive Predictive Value, and Negative Predictive Value*. Medicina (Kaunas). 2021 May;57(5):503. DOI: 10.3390/medicina57050503. PMCID: PMC8156826. PMID: 34065637.

[37] A. Johnning, E. Kristiansson, J. Fick, B. Weijdegård, D.J.G. Larsson. *Resistance Mutations in gyrA and parC are Common in Escherichia Communities of both Fluoroquinolone-Polluted and Uncontaminated Aquatic Environments*. Front Microbiol. 2015;6. doi: 10.3389/fmicb.2015.01355. Available from: https://www.frontiersin.org/journals/microbiology/articles/10.3389/fmicb.2015.01355

[38] M. Woegerbauer, J. Zeinzinger, B. Springer, P. Hufnagl, A. Indra, I. Korschineck, *et al. Prevalence of the aminoglycoside phosphotransferase genes aph(3)-IIIa and aph(3)-IIa in Escherichia coli, Enterococcus faecalis, Enterococcus faecium, Pseudomonas aeruginosa, Salmonella enterica subsp. enterica and Staphylococcus aureus isolates in Austria*. J Med Microbiol. 2014;63(2).

[39] M. Rezazadeh, H. Baghchesaraei, A. Peymani. *Plasmid-Mediated Quinolone-Resistance (qnr) Genes in Clinical Isolates of Escherichia coli Collected from Several Hospitals of Qazvin and Zanjan Provinces*, Iran. Osong Public Health Res Perspect. 2016;7(5):307-312. doi: https://doi.org/10.1016/j.phrp.2016.08.003. Available from: https://www.sciencedirect.com/science/article/pii/S2210909916300984

[40] R. Dorantes-Gilardi, L. Bourgeat, L. Pacini, L. Vuillon, C. Lesieur. *In proteins, the structural responses of a position to mutation rely on the Goldilocks principle: not too many links, not too few*. Phys Chem Chem Phys. 2018;20(39):25399-25410. doi:10.1039/C8CP04530E. Available from: http://dx.doi.org/10.1039/C8CP04530E

[41] L.M. Lim, N. Ly, D. Anderson, J.C. Yang, L. Macander, A. Jarkowski, A. Forrest, J.B. Bulitta, B.T. Tsuji. *Resurgence of Colistin: A Review of Resistance, Toxicity, Pharmacodynamics, and Dosing*. Pharmacotherapy. 2010;30(12):1279-1291. doi:10.1592/phco.30.12.1279.

A

Appendix 1

Table A.1: Recorded sensitivity for each model- and antibiotic in the study for easy fine-tuning (E). The model names are listed as the number of encoder blocks followed by the masking percentage for the type of fine-tuning- and pre-training used. The type of pre-training was either easy (E), hard (H) or no pre-training (N).

Ab name	1E-N	1E-E	1E-H	3E-N	3E-E	3E-H	12E-N	12E-E	12E-H
AMC	0.025	0.025	0.050	0.238	0.325	0.388	0.438	0.438	0.488
AMP	0.832	0.813	0.857	0.833	0.875	0.891	0.857	0.873	0.895
AZT	0.696	0.625	0.696	0.750	0.750	0.786	0.714	0.732	0.821
CZL	0.764	0.734	0.779	0.769	0.814	0.794	0.744	0.799	0.829
CPM	0.179	0.205	0.291	0.376	0.504	0.504	0.538	0.590	0.709
CTX	1.000	1.000	1.000	0.994	0.987	0.981	0.974	0.987	0.968
CXT	0.518	0.430	0.439	0.623	0.675	0.535	0.667	0.667	0.702
CPD	0.963	0.963	1.000	1.000	1.000	1.000	1.000	1.000	1.000
CTZ	0.682	0.611	0.753	0.702	0.778	0.783	0.712	0.758	0.798
CTR	0.741	0.737	0.777	0.785	0.839	0.834	0.741	0.805	0.878
CMP	0.041	0.245	0.265	0.429	0.551	0.673	0.592	0.694	0.796
CIP	0.772	0.873	0.826	0.868	0.886	0.873	0.886	0.886	0.921
ETP	0.000	0.000	0.000	0.000	0.000	0.000	0.154	0.154	0.154
GEN	0.362	0.329	0.396	0.577	0.483	0.530	0.584	0.758	0.725
IMI	0.000	0.000	0.000	0.000	0.000	0.000	0.149	0.286	0.286
LVX	0.889	0.959	0.930	0.965	0.982	0.965	0.965	0.959	0.977
MEM	0.000	0.000	0.000	0.000	0.000	0.000	0.149	0.429	0.429
STR	0.783	0.807	0.843	0.873	0.922	0.886	0.861	0.896	0.896
SXT	1.000	0.976	1.000	1.000	1.000	0.976	1.000	1.000	0.976
INN	0.836	0.796	0.829	0.855	0.875	0.862	0.875	0.882	0.875
TET	0.848	0.861	0.842	0.879	0.854	0.844	0.861	0.840	0.855
TOB	0.096	0.220	0.341	0.659	0.585	0.634	0.537	0.683	0.756
TRI	0.781	0.625	0.906	0.936	0.968	0.938	0.938	0.781	1.000
TMP	0.699	0.668	0.695	0.767	0.803	0.756	0.789	0.892	0.874

Table A.2: Recorded sensitivity for each model- and antibiotic in the study for hard fine-tuning (H). The model names are listed as the number of encoder blocks followed by the masking percentage for the type of fine-tuning- and pre-training used. The type of pre-training was either easy (E), hard (H) or no pre-training (N)..

Ab name	1H-N	1H-E	1H-H	3H-N	3H-E	3H-H	12H-N	12H-E	12H-H
AMC	0.000	0.000	0.000	0.125	0.200	0.088	0.338	0.288	0.338
AMP	0.781	0.783	0.795	0.770	0.857	0.821	0.804	0.799	0.826
AZT	0.589	0.536	0.571	0.714	0.679	0.696	0.643	0.643	0.750
CZL	0.709	0.739	0.729	0.734	0.789	0.784	0.729	0.754	0.769
CPM	0.111	0.051	0.137	0.256	0.282	0.325	0.282	0.402	0.564
CTX	1.000	1.000	1.000	1.000	1.000	1.000	0.994	0.968	0.974
CXT	0.281	0.175	0.298	0.395	0.500	0.447	0.518	0.412	0.553
CPD	1.000	1.000	0.963	0.963	1.000	1.000	0.926	1.000	0.926
CTZ	0.551	0.510	0.561	0.586	0.657	0.652	0.652	0.591	0.763
CTR	0.673	0.693	0.712	0.732	0.766	0.790	0.698	0.766	0.824
CMP	0.041	0.061	0.184	0.286	0.510	0.490	0.429	0.653	0.531
CIP	0.798	0.759	0.768	0.798	0.890	0.860	0.785	0.829	0.864
ETP	0.000	0.000	0.000	0.000	0.000	0.000	0.076	0.231	0.154
GEN	0.161	0.154	0.221	0.342	0.396	0.362	0.409	0.490	0.456
IMI	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.143	0.286
LVX	0.901	0.912	0.906	0.912	0.959	0.959	0.912	0.918	0.942
MEM	0.000	0.000	0.000	0.000	0.000	0.000	0.143	0.143	0.286
STR	0.729	0.741	0.771	0.837	0.873	0.867	0.849	0.867	0.855
SXT	1.000	0.952	0.976	0.952	1.000	1.000	1.000	1.000	0.976
INN	0.684	0.651	0.697	0.730	0.842	0.822	0.816	0.829	0.835
TET	0.775	0.754	0.770	0.826	0.802	0.824	0.748	0.799	0.840
TOB	0.121	0.073	0.073	0.341	0.390	0.439	0.341	0.512	0.707
TRI	0.875	0.906	0.936	0.938	0.938	0.938	0.906	0.906	1.000
TMP	0.596	0.601	0.601	0.704	0.758	0.758	0.704	0.767	0.789

Specificity tabeller:

Table A.3: Recorded specificity for each model- and antibiotic in the study for easy fine-tuning (E). The model names are listed as the number of encoder blocks followed by the masking percentage for the type of fine-tuning- and pre-training used. The type of pre-training was either easy (E), hard (H) or no pre-training (N)..

Ab name	1H-N	1H-E	1H-H	3H-N	3H-E	3H-H	12H-N	12H-E	12H-H
AMC	0.998	1.000	1.000	0.998	0.991	0.994	0.995	0.997	0.991
AMP	0.866	0.863	0.804	0.908	0.834	0.850	0.912	0.920	0.896
AZT	0.911	0.936	0.950	0.928	0.933	0.958	0.963	0.973	0.953
CZL	0.807	0.832	0.819	0.863	0.826	0.832	0.863	0.891	0.844
CPM	0.970	0.966	0.931	0.937	0.916	0.912	0.927	0.929	0.909
CTX	0.000	0.000	0.000	0.000	0.062	0.062	0.281	0.438	0.344

Continued on next page

Table A.3 – continued from previous page

Ab name	1E-N	1E-E	1E-H	3E-N	3E-E	3E-H	12E-N	12E-E	12E-H
CXT	0.948	0.935	0.935	0.925	0.935	0.934	0.971	0.950	0.951
CPD	0.000	0.417	0.250	0.667	0.667	0.583	0.750	1.000	0.750
CTZ	0.848	0.850	0.838	0.894	0.896	0.888	0.909	0.929	0.909
CTR	0.903	0.914	0.906	0.928	0.931	0.932	0.953	0.964	0.928
CMP	0.998	0.983	0.987	0.985	0.983	0.985	0.987	0.990	0.980
CIP	0.994	0.977	0.975	0.978	0.979	0.986	0.983	0.986	0.973
ETP	1.000	1.000	1.000	1.000	1.000	1.000	0.997	0.998	0.998
GEN	0.982	0.967	0.962	0.968	0.980	0.977	0.988	0.981	0.985
IMI	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.998	1.000
LVX	0.979	0.933	0.933	0.954	0.964	0.959	0.979	0.979	0.959
MEM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999
STR	0.926	0.924	0.914	0.907	0.951	0.963	0.934	0.951	0.961
SXT	0.613	0.806	0.742	0.645	0.871	0.903	0.871	0.903	0.806
INN	0.908	0.908	0.905	0.915	0.931	0.938	0.945	0.969	0.955
TET	0.858	0.890	0.804	0.852	0.917	0.909	0.826	0.935	0.934
TOB	1.000	0.998	0.996	0.977	0.990	0.992	0.988	0.985	0.985
TRI	0.707	0.780	0.707	0.707	0.829	0.780	0.805	0.854	0.854
TMP	0.927	0.950	0.943	0.945	0.959	0.961	0.969	0.977	0.963

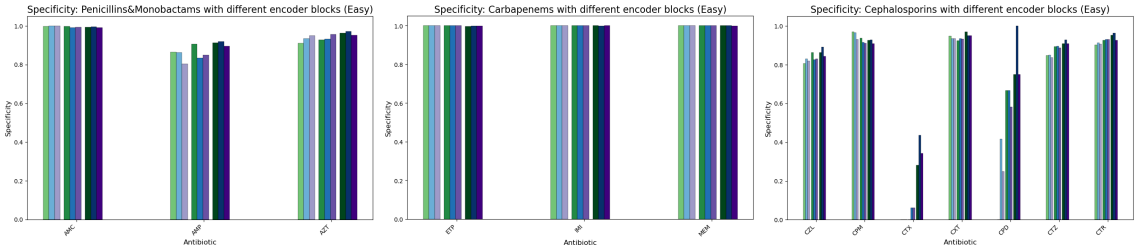
Table A.4: Recorded specificity for each model- and antibiotic in the study for hard fine-tuning (H). The model names are listed as the number of encoder blocks followed by the masking percentage for the type of fine-tuning- and pre-training used. The type of pre-training was either easy (E), hard (H) or no pre-training (N)..

Ab name	1H-N	1H-E	1H-H	3H-N	3H-E	3H-H	12H-N	12H-E	12H-H
AMC	1.000	1.000	1.000	1.000	0.997	0.998	0.999	0.996	0.990
AMP	0.816	0.838	0.727	0.793	0.690	0.852	0.884	0.909	0.684
AZT	0.903	0.946	0.938	0.946	0.921	0.943	0.975	0.955	0.948
CZL	0.794	0.816	0.841	0.804	0.819	0.794	0.882	0.894	0.816
CPM	0.966	0.983	0.981	0.944	0.929	0.940	0.966	0.925	0.901
CTX	0.000	0.000	0.000	0.000	0.000	0.000	0.063	0.219	0.094
CXT	0.943	0.968	0.931	0.938	0.892	0.899	0.919	0.956	0.926
CPD	0.000	0.000	0.333	0.250	0.417	0.000	0.250	0.750	0.417
CTZ	0.802	0.823	0.834	0.854	0.861	0.852	0.909	0.927	0.861
CTR	0.901	0.901	0.905	0.921	0.891	0.900	0.950	0.936	0.889
CMP	0.997	0.998	0.995	0.987	0.973	0.978	0.988	0.987	0.975
CIP	0.982	0.976	0.971	0.976	0.960	0.967	0.981	0.982	0.976
ETP	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
GEN	0.987	0.975	0.967	0.973	0.964	0.974	0.970	0.980	0.971
IMI	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
LVX	0.926	0.908	0.921	0.941	0.913	0.926	0.959	0.956	0.946
MEM	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.999	0.998
STR	0.924	0.956	0.936	0.909	0.946	0.924	0.956	0.961	0.966

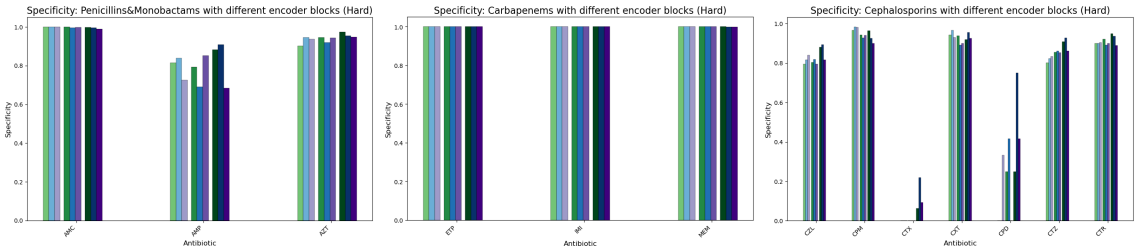
Continued on next page

Table A.4 – continued from previous page

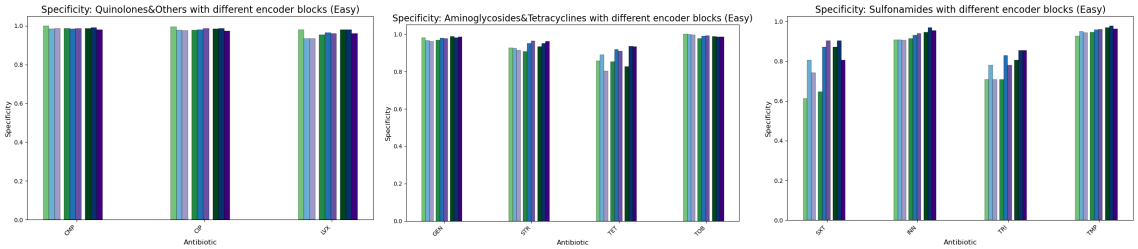
Ab name	1E-N	1E-E	1E-H	3E-N	3E-E	3E-H	12E-N	12E-E	12E-H
SXT	0.419	0.806	0.613	0.677	0.645	0.710	0.871	0.903	0.774
INN	0.924	0.917	0.898	0.910	0.910	0.914	0.924	0.941	0.941
TET	0.849	0.841	0.864	0.663	0.869	0.858	0.888	0.881	0.885
TOB	1.000	1.000	0.998	0.992	0.969	0.990	0.988	0.985	0.961
TRI	0.683	0.780	0.780	0.683	0.659	0.780	0.829	0.878	0.878
TMP	0.940	0.950	0.942	0.947	0.947	0.944	0.965	0.961	0.950



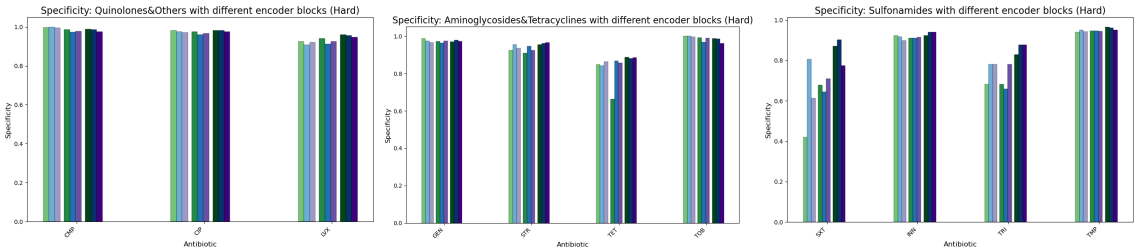
(a) Penic.&Monob., Easy. **(b)** Carbapenems, Easy **(c)** Cephalosporins, Easy.



(d) Penic.&Monob., Hard. **(e)** Carbapenems, Hard **(f)** Cephalosporins, Hard.

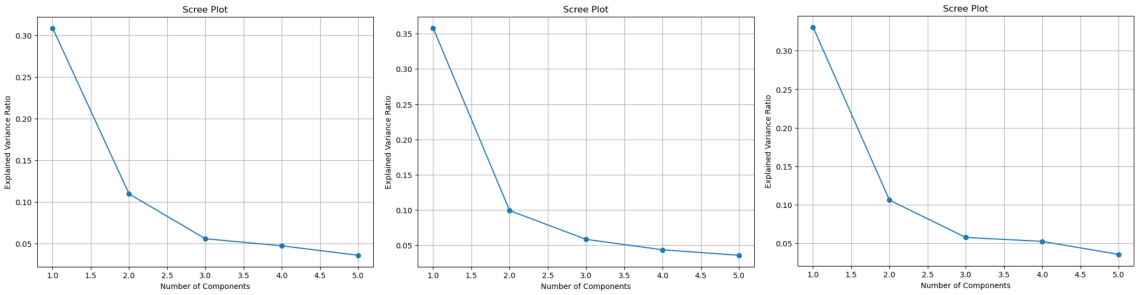


(g) Quino.&Others, Easy. **(h)** Amin.&Tetra., Easy . **(i)** Sulfonamides, Easy.

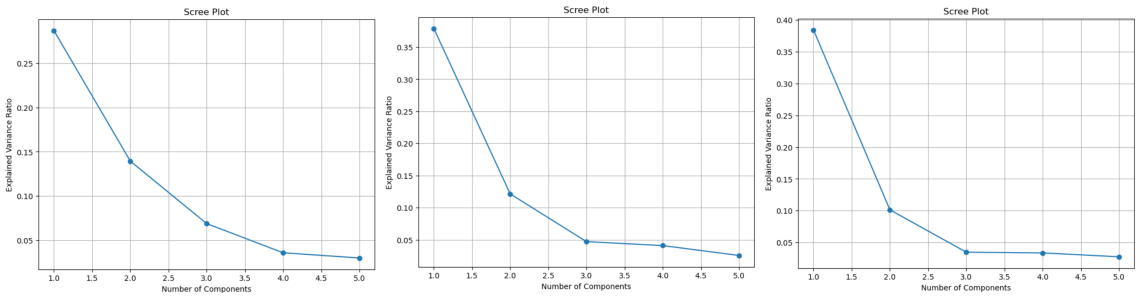


(j) Quino.&Others, Easy. **(k)** Amin.&Tetra., Hard . **(l)** Sulfonamides, Hard.

Figure A.1: Barplots with specificity data, grouped based on antibiotic classes. Each bar is coloured based on the type of run that was performed: No pre-training (green), Easy pre-training (blue) and Hard pre-training (purple). The saturation of colours indicates the complexity of the model: 1 encoder block (light), 3 encoder blocks (medium) and 12 encoder blocks (dark). Each subplot is labelled with which type of antibiotic class is present and whether the fine-tuning was done with an easy- or hard masking difficulty.



(a) Easy FT, No PT. **(b)** Easy FT, Easy PT. **(c)** Easy FT, Hard PT.



(d) Hard FT, No PT. **(e)** Hard FT, Easy PT. **(f)** Hard FT, Hard FT.

Figure A.2: Scree plots for the PCA-CLS analysis for all selected models. Each subplot is captioned with the settings used in training to produce the model.

DEPARTMENT OF MATHEMATICAL SCIENCES
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY