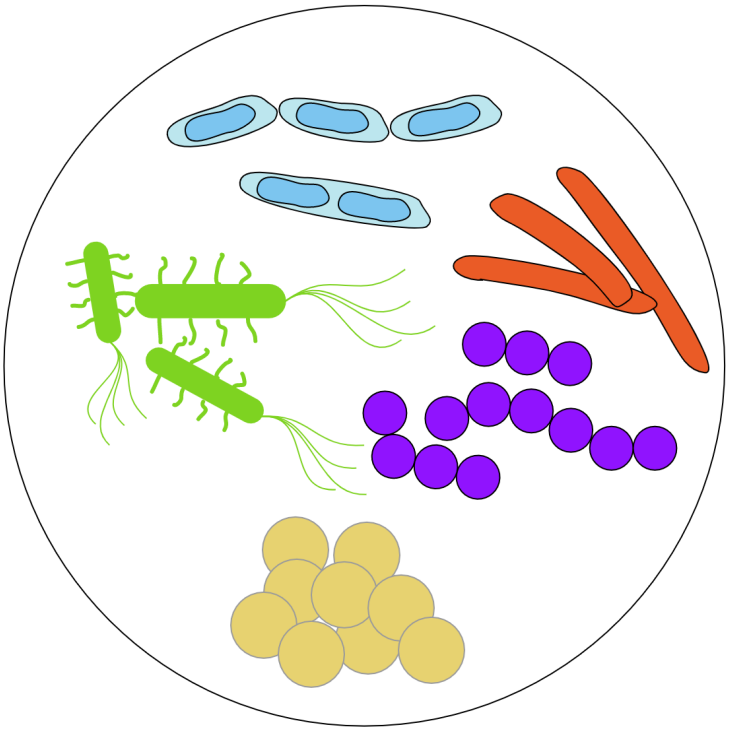
CHALMERS
UNIVERSITY OF TECHNOLOGY

Taxonomic Classification of Metagenomic Short Reads

Evaluation of Metagenomic Analysis Software and a Neural Network Approach to Improving Read Classification



Master's thesis in Complex Adaptive Systems

MATILDA WIKSTRÖM

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2020

Master's thesis 2020

Taxonomic Classification of Metagenomic Short Reads

Evaluation of Metagenomic Analysis Software and a Neural Network Approach to Improving Read Classification

MATILDA WIKSTRÖM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Mathematical Sciences
Chalmers University of Technology
Gothenburg, Sweden 2020

Taxonomic Classification of Metagenomic Short Reads -
Evaluation of Metagenomic Analysis Software and a Neural Network Approach to Improving Read Classification
MATILDA WIKSTRÖM

Supervisors: Fredrik Dyrkell & Suvash Thapaliya, 1928 Diagnostics
Examiner: Erik Kristiansson, Department of Mathematical Sciences

Master's Thesis 2020
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Illustration of a metagenomic sample.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2020

Taxonomic Classification of Metagenomic Short Reads -
Evaluation of Metagenomic Analysis Software and a Neural Network Approach to Improving Read Classification
MATILDA WIKSTRÖM
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Hospital acquired infections is a large issue in modern healthcare and they are becoming more difficult to treat due to increasing antibiotic resistance. To limit the spread of serious bacterial infections there is a need for fast diagnosis and treatment. The advent of next-generation sequencing has drastically reduced sequencing costs making it feasible to analyze metagenomic samples taken directly from the patients.

This thesis has evaluated three metagenomic analysis tools with regards to species identification and abundance estimation for simulated metagenomic short reads originating from 15 different species. All tools showed different strengths and weaknesses, however an outstanding weakness found was classification of reads belonging to the *Streptococcus mitis* group and the *Mycobacterium tuberculosis* complex.

To improve the classification of reads from *Streptococcus* and *Mycobacterium* we implemented a feed-forward neural network. For *Streptococcus* species we obtained an accuracy of 95% while our models failed to reach higher than 31% accuracy for *Mycobacterium* species. One of the causes for these different results is that the pairwise BLAST identity within the species groups are around 95% similarity for *Streptococcus* and 99% for *Mycobacterium*.

Keywords: Metagenomics, Machine Learning, Neural Network, Taxonomic Classification

Acknowledgements

I would like to direct a thank you to my supervisors Fredrik Dyrkell and Suvash Thapaliya. This project would not have been what it is without the support and guidance you have provided throughout my thesis work.

Secondly, I want to thank Emil Carlsson, PhD student at Chalmers, for introducing me to 1928 Diagnostics. I appreciate that you always only been a message away for interesting discussions and feedback.

Finally, a massive thank you to the entire team at 1928 Diagnostics for welcoming me to the 1928 family. I will truly miss being a part of your fantastic team and hope that our paths will cross soon again.

Matilda Wikström, Gothenburg, May 2020

Contents

List of Figures

List of Tables

1

Introduction

Every year millions of people fall ill in infectious diseases [3]. Out of these, a substantial number are so called *hospital acquired infections*, meaning the patient was infected while admitted at the hospital. At the same time, many bacterial species are evolving and develop resistance towards many of our most powerful antibiotics [4]. As one can understand, this pose a huge problem in today's healthcare and there is a great need to limit the spreading these infections. An essential part to reducing the spread of diseases is to identify the source of the infection and treating those who already have severe bacterial infections. To be able to give optimal treatment to these patients it is important to quickly identify and characterize the causing pathogen, increasing the chance of recovery.

Current methods for diagnosing often requires cultivation of bacteria which can be a very time consuming procedure [5]. In addition to this, it is only possible to cultivate about 1% of known microorganisms which leads to a large amount of information that can be used for establishing a diagnosis is being overlooked [6]. With the affordable sequencing techniques that are available it is possible to bypass the cultivation step and instead analyze the entire raw sample, called a *metagenomic* sample, directly [7].

Metagenomic analysis has great potential in clinical healthcare as it can provide new information that was previously missed, however the first step is to understand the composition of the communities that can be found in the samples. In order to do so, it is needed to determine what species are present in the sample by performing *taxonomic classification* as well identifying the relative amount of DNA belonging to each of the identified species, referred to as *abundance estimation* [6].

Sequencing of metagenomic samples generates millions short reads which comes with some challenges in how to handle and analyze the data [8]. Over the years many metagenomic analysis software have emerged with the goal of classifying metagenomic data to be able to understand the behaviour of microbial communities. As the underlying algorithm differs between the software the performance of them may also differ, depending on the sample being analyzed. Therefore it is necessary to evaluate the performance of specific methods on the type of data it is intended to be used on [9].

2

Project Aim

The aim of this thesis is to investigate different methods for species identification and abundance estimation of bacterial metagenomic samples. This thesis will also investigate how neural networks can be applied to improve the taxonomic classification. The aim can therefore be divided into two parts:

I. Through experiments evaluate strengths and weaknesses of established metagenomic analysis software.

II. Investigate if neural network models can improve performance in the areas where established methods are lacking in performance.

3

Theory

3.1 Biological Background

In this section we aim to give an introduction to the biological background needed to understand the concepts *species identification* and *metagenomic sample*. To get an understanding for species identification we start by briefly describe what taxonomy is and the hierarchical structure that species are classified according to. Following that we introduce DNA and how it can be analyzed in order to learn about a species characteristics. Finally we give an introduction to metagenomics and how it is currently analyzed from a species identification perspective.

3.1.1 Taxonomy

Researcher are continuously discovering new species and current estimates suggests that there may be as much as 100 million different species on earth [10]. In order to keep track of the relationship between species a biological filing system is used and as of today approximately 2 million species have been described and catalogued [11]. The science of classification of species is called *taxonomy* and the taxonomic tree is divided into several levels. There are many ways to define the taxonomic tree, however commonly it is divided into the seven levels shown in figure 3.1. These levels are organized as "boxes in a box", meaning the broader categories are successively divided into the more narrow categories creating a *taxonomic hierarchy*. Here *kingdom* is the broadest category and each of the kingdoms are split into multiple groups in the *phylum* level. Further, each phylum is split into several *classes*, and so on until we reach the more specific levels *genus* and finally *species* [1, p.3-8]. Traditionally, organisms are classified and ordered in the taxonomic tree according to their characteristics. However, for prokaryotes (Bacteria and Archaea) it is difficult to accurately distinguish different species only based on their characteristics and therefore it is necessary to consider other features such as genetic structure [12].

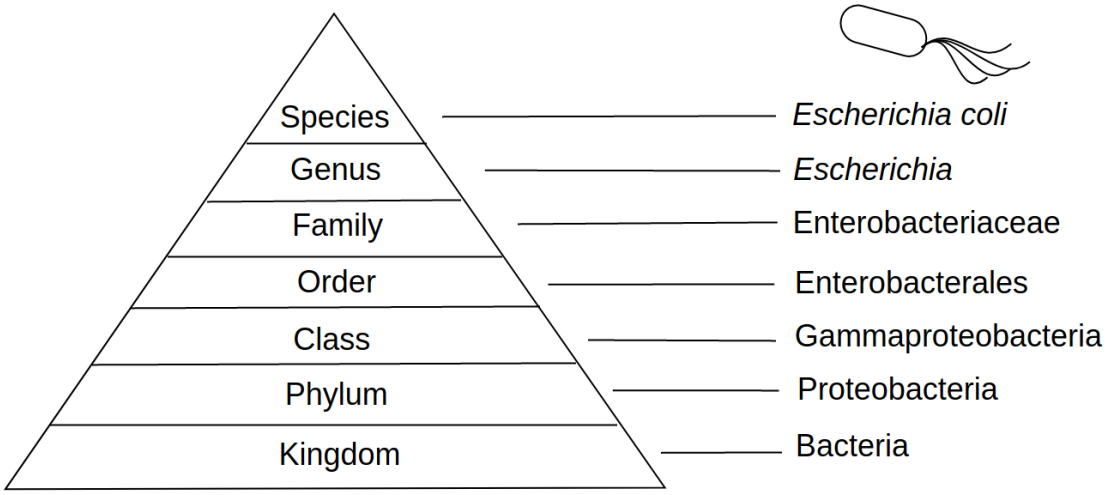


Figure 3.1: Illustration of the taxonomic hierarchy based on the taxonomic levels according to Julian Sutton's book *Biology* [1]. To the right an example of the taxonomic classification of the bacteria *Escherichia coli* is shown.

Sometimes there are groups of species within a genus that are more similar than usual. In these cases an additional, informal level between *genus* and *species* can be introduced to highlight their relation. An example of such a group is the *Streptococcus mitis* group. A number *Streptococcus* bacteria belong to this group that are genetically similar but have different characteristics as some are highly pathogenic while others are commensal and occurs without hurting the host [13].

3.1.2 DNA

All living organisms contain a large amount information on how to build and maintain that specific organism. The entire genetic material in an organism is called the *genome* and is stored in complex molecules called *Deoxyribonucleic acid*, more known as *DNA*. The DNA molecule takes the form of a double helix, figure 3.2, where the backbone structure is made of sugar and phosphate molecules. The two helix shaped structures are bonded together by two *nucleotides* referred to as a base pair (bp). The four nucleotides that create these base pairs in DNA are: *adenine* (A), *cytosine* (C), *guanine* (G), and *thymine* (T) and can be seen as the the biological alphabet [14, p. 8-9].

The functions and characteristics of an organism is governed by proteins that are expressed within the cells. The instructions for assembling the proteins is encoded in the DNA and the entire DNA contains instructions for creating a wide range of different proteins. The part of the DNA that contains the information for creating a specific protein is called a *gene* [14, p. 8-9].

In 1859 Darwin introduced the concept of evolution, the process of how organisms change over many generations. Sometimes when cells in an organism divide events happen that change the genetic composition of the cell, such events can be mutations or genetic recombination. These changes can result in new characteristics of the organism and as changes are passed on to later generations over long time, new species can evolve [15, p. 42].

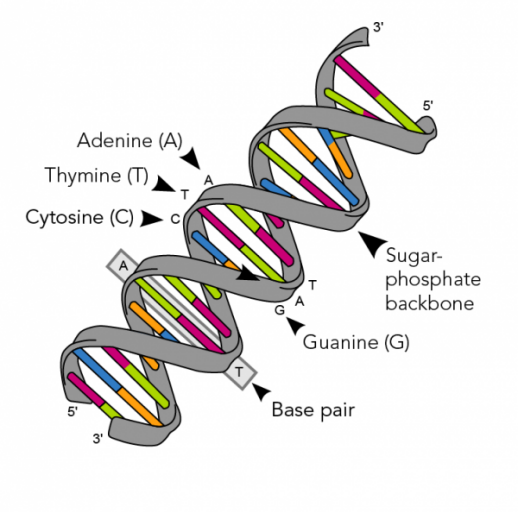


Figure 3.2: Double helix structure of the DNA molecule showing the sugar-phosphate backbone, nucleotides, and the base pair. Image from *Let's Talk Science* under license CC BY-SA 3.0 [2].

3.1.3 Analyzing DNA

Knowing that characteristics of a species is encoded in the DNA opens up a wide range of possibilities for understanding what causes diseases and how they should be treated. With a fast paced development of technology the techniques for DNA sequencing have improved greatly improved in the last years. The advent of *Next Generation Sequencing* (NGS) has made it possible to sequence and analyze DNA in a fast and affordable way, giving us new insights in the field of *genomics*. NGS consists of a range of sequencing techniques for creating a digital representation of the sequenced genome. However, the common factor for all NGS techniques is that they can sequence DNA fragments in parallel, generating a lot of data in a short time [16]. The sequenced DNA fragments are called *reads* and can be of varying lengths and quality depending on the technique used, though *short read* techniques commonly give sequence lengths of 50-400 bp [17].

In bioinformatics *k-mers* are often considered for analyzing DNA sequences where a k-mer is a subsequence of length k. Given k as well as the four nucleotides (A, C, G and T) there exists 4^k unique k-mers. While a larger k gives makes it easier to identify differences between sequences it also has a higher computational cost. By considering overlapping k-mers it is possible to create a distribution of k-mers over a longer sequence giving a "fingerprint" that can be used for analysis. Figure 3.3 demonstrates how to extract 3-mers out of a sequence.

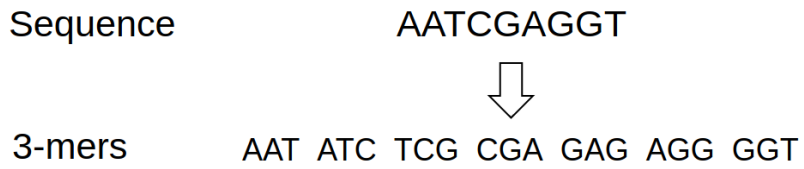


Figure 3.3: Demonstration of how overlapping 3-mers are created.

3.1.4 Metagenomics

With the entrance of NGS and improving sequencing techniques it has allowed the field of *metagenomics* to grow stronger. In "regular" genomics one is limited to studying samples containing only one species, while metagenomics aim to study samples that contain a mix of many species. Through *metagenomic shotgun sequencing* (MGS), where the DNA is fragmented into shorter segments before sequencing, is it possible to analyze raw environmental samples and study the composition of communities containing both previously known and unknown organisms [18]. Metagenomics has shown that a small sample can contain thousands of organisms, many of which was missed when culture based methods was used for analysis. Following that, one of the key questions within metagenomics is *Who is there?* and this question can be answered by identifying present organisms and their relative abundance in a sample [6].

However, a limitation with metagenomic sequencing is that it does not only capture bacterial DNA, but also all redundant DNA from the human genome. The issue here is that the large amount of human DNA may overshadow the lower levels of pathogenic DNA that is of interest. Therefore, current diagnosis methods often relies on isolation and cultivation of colonies of a single bacterial species. This task is both very time consuming and requires a priori knowledge in regards to what bacteria one is looking for, but ensure there is enough bacterial DNA in order to perform accurate analysis [5]. If the identification of organisms can be done accurately using metagenomics, there is much to be gained from a clinical perspective. For example, it could be possible to directly sequence blood samples from sepsis patients and hence shorten the time to identify what pathogen that is causing the infection and start the correct treatment.

As metagenomics are gaining in popularity, the number of analysis software are increasing. Many of the current tools for taxonomic classification are based on a *lowest common ancestor* (LCA) approach. With this approach the reads, or some representation of them, are compared towards a database and mapped to all possible species where this representation occurs. Some reads may be matched to more than one species and if that is the case, the read will instead be assigned to a lower level in the taxonomic tree which would give an unambiguous result for that specific read. Due to that the reads can be classified at different taxonomic levels, i.e. some reads are classified at species level, other at genus level etc. [19].

3.2 Machine Learning

Machine learning is a field within artificial intelligence (AI) where a program aims to perform a task without being explicitly told how to perform it. Instead a data-driven approach is used where the models are designed such that they can learn complex patterns from a set of data, gaining general knowledge about the task it is set to perform [20]. In machine learning two typical tasks are *regression* and *classification*. In regression we aim to model the relation between input and output variables and this type of models are often used in forecasting problems. For classification tasks are we, instead of predicting a specific value, assigning the data point to one of multiple predefined categories [21, p. 97-101].

For both regression and classification it is common to train the models on example data where the correct output value or class is known. If we allow the model to use the correct answers, *labels*, during the learning process we refer to this as *supervised learning*. In this project we will implement a classification algorithm and supervised learning for classification can be described as follows.

Consider an *input space* $\mathbf{X}$ and its corresponding *output space* $\mathbf{Y}$. The relation between the input space and the output space is given by a function

$$g : \mathbf{X} \to \mathbf{Y}, \tag{3.1}$$

which we want to identify. However, finding g without knowing anything about the space g lies in is an almost impossible task. Therefore machine learning tries to find a function f in some *function space* F which approximates g such that a *loss function* $\mathcal{L}$ is minimized.

Assuming we have a training set $\mathbf{x} \subset \mathbf{X}$ with labels $\mathbf{y} \subset \mathbf{Y}$ the optimization problem becomes

$$\min_{f \in F} \mathcal{L}(f(\mathbf{x}), \mathbf{y}). \tag{3.2}$$

The choice of loss function depends on the task being solved. Cross-entropy loss given by equation (3.3) is commonly used for multi-class classification tasks [22, p. 395]

$$\mathcal{CE}(f(\mathbf{x}), \mathbf{y}) \;=\; - \sum_{(\mathbf{x},\mathbf{y}) \in N} \mathbf{y}^T log(f(\mathbf{x})). \tag{3.3}$$

3.2.1 Artificial Neural Network

Artificial neural networks (ANN) is a method inspired by the biological neural network found in human's and other animal's brains. The brain consists of a complex structure of connected neurons which together can carry out advanced tasks such as image recognition or movement of body parts [23]. Going forward the term *neural network* will refer to the artificial neural network.

In machine learning, a neural network is defined as a graph $G = (V, E)$ where V is the nodes, here referred to as *neurons*, and E the edges connecting the nodes. Each edge has its individual weight and the edges in the neural network of therefore often referred to as *weights*, connecting the nodes. The state of a neuron j, h_j, is given by the the weighted sum of input signals, x_i, with weights, w_i, and a *bias term*, b_j, according to

$$h_i \ = \ \sum_j w_{j,i} x_i \ + \ b_i. \tag{3.4}$$

The output signal, y_j, is further given by applying an *activation function*, σ, to the state of the neuron,

$$y_i \ = \ \sigma(h_i). \tag{3.5}$$

The output for the entire network can be written on matrix form as

$$\hat{\mathbf{y}} \ = \ \sigma(\mathbf{W}\mathbf{x} + \mathbf{b}). \tag{3.6}$$

3.2.1.1 Feed Forward Neural Network

The most basic structure of a neural network is the *feed forward* network referring to a directed neural network that does not feed any of the intermediate values back to network. In order to approximate the function we are looking for the feed forward neural network connects multiple functions in a chain structure. In practice each of these functions are represented by a *layer* of neurons. All networks consists of an input and an output layer but can have an arbitrary number of layers in between these, so-called *hidden layers* [21, p. 164-167]. Figure 3.4 shows the structure of a feed forward neural network with one hidden layer.

In a more general case we can have a network with L layers. Then, for the hidden layer l, the input is given by the output from the previous layer, $\mathbf{z}^{(l-1)}$. The function which this single layer represents is

$$f^{(l)} = \sigma(\mathbf{W}^{(l-1,l)}\mathbf{z}^{(l-1)} + \mathbf{b}^{(l)}). \tag{3.7}$$

Connecting all L layers in the network then gives us the resulting output

$$\hat{\mathbf{y}} = f^{(L)}(f^{(L-1)}(...f^{(1)}(\mathbf{x})). \tag{3.8}$$

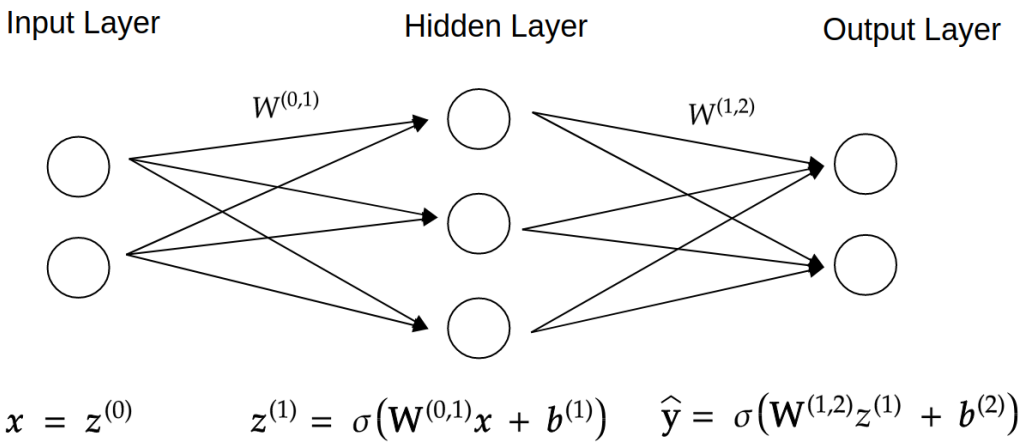


Figure 3.4: Structure for a feed forward neural network with one hidden layer. Below are the calculations performed on the input vector $\mathbf{x}$ to obtain the output of the network.

3.2.1.2 Training of Neural Networks

As mentioned above, training a neural network is an optimization problem where we want to minimize the loss function. In order to do so, the *back-propagation* algorithm is widely used to compute the gradients for each layer which are then updated using the *gradient descent* optimization technique. Back-propagation calculates the gradients of the loss function with respect to each individual parameter one layer at the time starting at the output layer. By recursively applying the chain-rule the gradient for earlier layers can be calculated efficiently without unnecessary calculations of re-occurring expressions in the intermediate steps [21, p. 200-209].

The derivatives of the loss with respect to the weights and biases in the step from layer $l-1$ to l are

$$\partial f_{\mathbf{W}}^{(l)} = \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l-1,l)}}, \tag{3.9}$$

$$\partial f_{\mathbf{b}}^{(l)} = \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}}. \tag{3.10}$$

These derivatives are further used in the gradient based update step to update weights and biases according to

$$\mathbf{W}_{t+1}^{(l-1,l)} = \mathbf{W}_{t}^{(l-1,l)} - \boldsymbol{\eta}_{t}^{(l-1,l)} \odot \boldsymbol{\delta}_{t}^{(l)} (\mathbf{z}_{t}^{(l-1)})^{T}, \tag{3.11}$$

$$\mathbf{b}_{t+1}^{(l)} = \mathbf{b}_{t}^{(l)} - \boldsymbol{\eta}_{t}^{(l)} \odot \boldsymbol{\delta}_{t}^{(l)}. \tag{3.12}$$

Here $\boldsymbol{\eta}^{(l)}$ is the *learning rate* for the parameters in layer $l < L$ and $\odot$ is the element-

wise product. Further $\boldsymbol{\delta}_t^{(l)}$ is defined as

$$\boldsymbol{\delta}_t^{(L)} = \frac{\partial \mathcal{L}}{\partial f_{\mathbf{W},\mathbf{b}}^{(L)}(\mathbf{x})} \odot \sigma'(\mathbf{h}_t^{(L)}) \tag{3.13}$$

$$\boldsymbol{\delta}_t^{(l)} = (\mathbf{W}_t^{(l,l+1)})^T \boldsymbol{\delta}_t^{(l+1)} \odot \sigma'(\mathbf{h}_t^{(l)}) \tag{3.14}$$

In traditional gradient descent we forward propagate the entire dataset before calculating the gradient and update our parameters. This method has the advantage of always converging, either to the global minimum if the loss function is convex, or a local minimum for non-convex loss functions. The disadvantage on the other hand is that a lot of memory will be needed to store the gradients making it unsuitable for large datasets. On the other end of the spectrum we have stochastic gradient descent. Here the parameters are updated after each sample has been passed through the network. This is much faster than traditional gradient descent but due to the constant updates it will be difficult for the method to converge to the exact minimum. The midway between the above approaches is the so-called *mini-batch gradient descent*. Here, the update is performed after each mini-batch of m samples allowing us to chose a value on m such that the training is fast while also converging towards the minimum. This is normally the go-to method when training neural networks [24].

3.2.1.3 Overfitting and Regularization

When implementing neural networks it is easy to think the the larger the network, the better the performance. This is far from true as it has show that models large models are more prone to *overfitting*. Overfitting happens when our models stops learning the general features of our data and instead starts learning specific traits for the training data. The result of this is the model performs well on the training set, giving the impression of of being very good, while the performance on the validation set is much worse [25]. A simple way to detect overfitting is by comparing the loss values for the training and validation data. For a model that learns general feature the loss will decrease for both datasets, as seen up until epoch E in figure 3.5. From epoch E and forward we can then see the loss for the validation data is increasing at the same time as the loss for the training data is still decreasing, this is a clear sign of overfitting.

Fortunately there are ways to reduce overfitting in our models. The first thing to do, if possible, is to increase the number of samples in the training set. If there is enough data to train on it will be more difficult for the model to learn the specific features of the training data and therefore learns to generalize better. However, one does not always have access to more data and that is when regularization techniques as *weight regularization* and *dropout* comes in handy [25].

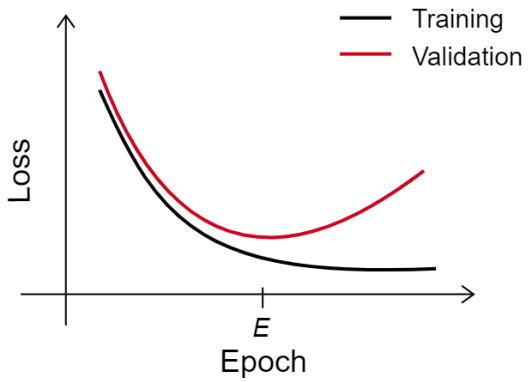


Figure 3.5: Loss curves for training and validation set showing that the model starts to overfit after epoch E.

With weight regularization we introduce a way to reduce the size of the weights in the network. This is done by adding an extra term to the loss function that penalize the weights each time they are updated, forcing their value towards zero. By doing this, some hidden neurons will become neglectable and the complexity of the model will be reduced. $L1$- or $L2$- regularization are the most common methods for weight regularization and work in the same way, the difference is the penalization term that is added to the loss function. The updated loss functions for $L1$ and $L2$ is given by equation (3.15) and (3.16) respectively and λ is the *regularization rate* [26]

$$\mathcal{L}_{L1}(\mathbf{W}, \mathbf{b}) = \mathcal{L}(\mathbf{W}, \mathbf{b}) + \lambda \sum_{i,j} |w_{i,j}|, \tag{3.15}$$

$$\mathcal{L}_{L2}(\mathbf{W}, \mathbf{b}) = \mathcal{L}(\mathbf{W}, \mathbf{b}) + \frac{\lambda}{2} \sum_{i,j} w_{i,j}^2. \tag{3.16}$$

The second popular regularization technique to use is dropout. The basic idea with dropout is to turn off some the neurons during training and by that temporarily making the network simpler. Once it is time for evaluation of the test data all neurons are turned back on again. What neurons that are being turned off is chosen at random with a probability $\mathbf{P}$ at each training step and therefore we will effectively have a different network architecture each time [27].

A demonstration of how dropout works is shown in figure 3.6. On the left we have a network with two hidden layers where no dropout as been applied and all neurons are active. On the right we have applied dropout with probability $\mathbf{P} = 0.5$ to the hidden layers, resulting in approximately half of the neurons being turned off during training, i.e. there will not be any update to the weights and biases associated with those neurons.

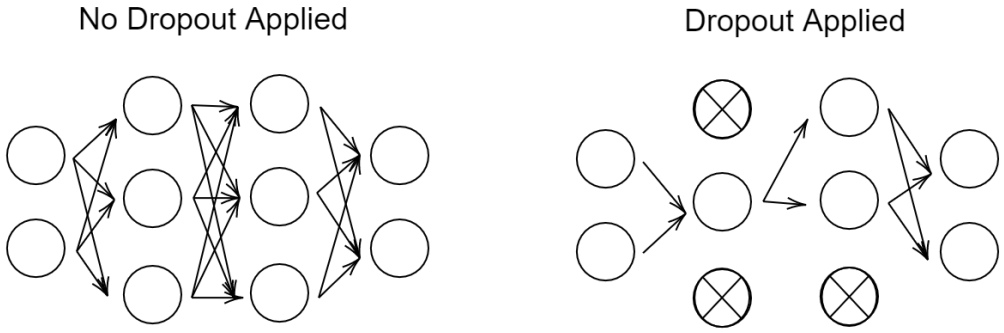


Figure 3.6: Illustration of the complexity to a neural network when dropout is not applied (left) compared to when dropout is applied (right) with probability **P**=0.5.

3.3 Data Processing

Using sentences and words as input to machine learning models have opened up a lot of new applications for machine learning. However, due to the fact that computers only can process numerical data it is necessary to transform the sequences from text to numbers before we can use them in our models. This section presents the theory for the methods that have been used to perform the transformation from text to numbers. In addition to this, we describe *principal component analysis* which was used to explore the data.

3.3.1 Bag-of-Words

Bag-of-words is a feature extraction method commonly used in *natural language processing* for representing text data. Given a set of documents a vocabulary of all known words is created and each word receives an index. For each document, the number of occurrences for each word is counted and stored in an array at the index corresponding to the word's index in the vocabulary. If a word is not present in the document, the count will naturally be set to zero. In this project the DNA sequences will represent the documents and the overlapping k-mers, as described in 3.1.3, are our "words" [28].

The bag-of-words model is simple to implement, however it comes with limitations. Representing text as a bag of words does not take the order of them into consideration, and therefore a lot of information stored in the context may be lost. Another negative aspect is that the scoring system will give more emphasis to words that are occurring often causing rare words, which may contain more relevant information, to be seen as less important. *Term Frequency - Inverse Document Frequency* is a strategy for giving rare words more importance in the representation [28].

3.3.2 Term Frequency - Inverse Document Frequency

Term Frequency - Inverse Document Frequency (TF-IDF) is a method for re-scoring the bag-of-word model to give the relative importance of each word. As the name suggests, there are two parts to this method; the Term Frequency and the Inverse Document Frequency [29].

The term frequency $tf(t, d)$ is given by how often a word t occurs in document $d \in D$. Since some documents may be longer than other, it is common to normalize this measure by dividing with the total number of words in the document. The term frequency then becomes

$$tf(t, d) \;=\; \frac{\text{number of times } t \text{ occurs in } d}{\text{total number of words in } d}. \tag{3.17}$$

Further, the inverse document frequency $idf(t, D)$ is a measure of how rare a word is across all documents D. If a word occurs in many documents we want to scale down the importance and it is present in few documents we want to scale up the importance. The inverse document frequency is given by

$$idf(t, D) \;=\; log\left(\frac{\text{total number of documents}}{\text{number of documents with word } t \text{ in it}}\right). \tag{3.18}$$

The TF-IDF score for a word is then given by simply multiplying the two parts

$$TF\text{-}IDF(t, d, D) \;=\; tf(t, d) \;\cdot\; idf(t, D). \tag{3.19}$$

3.3.3 Principal Component Analysis

Encoding text into numbers often transforms the data into a very high dimensional space. By using principal component analysis (PCA) it is possible to reduce the feature space while conserving as much relevant information as possible. Assume the data is represented in p dimensions which you want to reduce to m dimensions. As PCA aims to contain as much information as possible it will identify the m directions capturing as much variance in the data as possible. By finding the m largest eigenvalues and the corresponding eigenvectors, a new m-dimensional space can be created where the eigenvectors are the bases. Finally, the original, p-dimensional, data can be projected onto the new space transforming the data from p to m dimensions [30, p. 374-379].

4

Methods

In this section we will describe the work process of this thesis. The first part of the project aimed at investigating already available software for metagenomic analysis, however there was no predefined process for evaluating these. Therefore, the evaluation step is performed with inspiration from previously published studies [8, 31]. The generation of datasets, as well as metagenomic analysis software was run on Linux OS Ubuntu 18.04.

The second part aimed to improve some part of the metagenomic analysis using artificial neural networks. Implementation of these neural networks was done using Python 3.6 and the *PyTorch* framework. Training and evaluation was conducted on a 16-core Google Cloud instance. An overview of all software and packages used in the project is available in Appendix A.

4.1 Dataset

A key part to a successful project is the accessibility to a proper dataset. To be able to evaluate the performance of the tested methods we need a dataset where we know the composition of species. In addition to this, the training of the neural network models require us to know the label of each individual read. Such datasets have not been available for this project leading us to use simulated reads. The reads used for training and evaluation of methods have all been simulated using InSilicoSeq [32] which simulates reads corresponding to Illumina short reads [33]. There are several positive sides to using simulated data; besides getting the label for each read we also have the possibility to select what reference genomes to generate the reads from, what error model to be applied, and the exact abundance for each included species.

In this project a few different datasets, with different species combinations have been used. An outline of the species included in each dataset is presented in table 4.1. For the evaluation of metagenomic software the datasets listed as *1928miseq, 1928hiseq, Streptococcus,* and *Mycobacterium* was used. Moreover, for training and evaluation of the neural network models the datasets denoted as *Streptococcus, Mycobacterium,* and *Group* was used. All datasets were created to have uniform abundance among

the species and the MiSeq error model was used for all datasets except the *1928hiseq* dataset which had the HiSeq error model applied. In this project one reference genome per species has been used and all reference genomes were downloaded from *NCBI RefSeq* [34]. A list of the accession numbers for the used reference genomes is available in Appendix B.

1928miseq/1928hiseq	**Group**
Salmonella enterica	*Salmonella enterica*
Klebsiella pneumoniae	*Klebsiella pneumoniae*
Escherichia coli	*Escherichia coli*
Mycobacterium tuberculosis	*Mycobacterium tuberculosis*
Staphylococcus aureus	*Mycobacterium africanum*
Neisseria gonorrhoeae	*Mycobacterium bovis*
Enterococcus faecium	*Mycobacterium canettii*
Clostridioides difficile	*Streptococcus pneumonia*
	Streptococcus pseudopneumoniae
	Streptococcus mitis
	Streptococcus oralis
Streptococcus	**Mycobacterium**
Streptococcus pneumoniae	*Mycobacterium tuberculosis*
Streptococcus pseudopneumoniae	*Mycobacterium africanum*
Streptococcus mitis	*Mycobacterium bovis*
Streptococcus oralis	*Mycobacterium canettii*

Table 4.1: Species included in each type of dataset. Reads were simulated from one reference genome for each species and all datasets were created to have uniform abundance.

4.2 Evaluation of Existing Metagenomic Software

As mentioned in section 3.1.4 there are many metagenomic analysis software available. The selection of software to evaluate was based upon the what underlying method they are using, how often they occurred in other papers, and on recommendation from supervisors. Common for all software is that they use the LCA approach. *Kraken2* is a k-mer based method which for each k-mer in the sequence maps it to the lowest common ancestor for all genomes containing an exact match of the k-mer [35]. *Bracken* (Bayesian Reestimation of Abundance with KrakEN) is an extension for Kraken2 utilizing bayesian statistics to re-distribute reads assigned to a higher taxonomic level to species level in order to improve abundance estimation [36]. The third tool is *MetaPhlan2*, a marker based method identifying marker genes in the reads and using a reference database identifying species and estimating relative abundance in the dataset [37, 38].

To evaluate metagenomic analysis software there is unfortunately no well defined process on how it should be done as it usually depends on what one is interested in. Here, it was decided to focus on investigating how well Kraken2, Bracken and MetaPhlan2 succeeds in identifying the correct species, if there are additional species identified, and how well they can estimate the abundance of the included species. In order to do so, we have evaluated the software with respect to

I. **Sequencing technique:** We want to investigate if the error models corresponding to Illumina's *MiSeq* and *HiSeq* sequencing had any impact on the software's performance.

II. **Closely related species:** How well does the software manage to perform on a dataset containing many species that are genetically very similar? For this a simulated dataset with *Streptococcus* species all belonging to the *Streptococcus mitis* group was used.

III. **Prediction stability:** How consistent the predictions given by the software are. Will they predict the same results for multiple datasets containing the same species composition? To investigate this each type of dataset has been simulated and evaluated five times.

4.2.1 Quantifying Classification Performance

For species identification we are only interested in the presence of a species in the sample, hence we have a binary classification problem with the confusion matrix given by figure 4.1. To measure how well the software are at identifying the relevant species in our datasets we will use precision, recall and F1-score given by equations (4.1) - (4.3) with notations from the confusion matrix in figure 4.1.

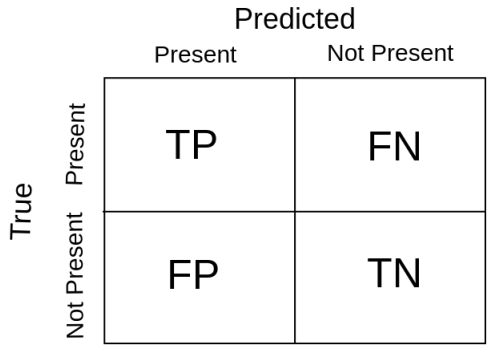


Figure 4.1: Confusion matrix for calculating precision, recall and F1-score to measure performance in identifying relevant species.

$$precision \ = \ \frac{TP}{TP \ + \ FP} \tag{4.1}$$

$$recall \ = \ \frac{TP}{TP \ + \ FN} \tag{4.2}$$

$$F1\text{-}score \ = 2 \ \cdot \ \frac{precision \ \cdot \ recall}{precision \ + \ recall} \tag{4.3}$$

The precision gives us how many of the identified species that are relevant while recall gives a measure of how good the method is at identifying the correct species. The F1-score is the harmonic mean of the precision and recall and gives us a way to measure the overall accuracy of the method with its best value being 1.

4.2.2 Evaluation of Relative Abundance Estimation

Evaluating how good the methods are at predicting the abundance we consider the predictions to be in a space of S dimensions, where S is the number of species available in our taxonomic definition. To measure the distance from the prediction, $\boldsymbol{p}$, to the true abundance, $\boldsymbol{t}$, we use the $L2$ - distance in equation (4.4). This measure was chosen as it has previously been used in the benchmarking study *Benchmarking Metagenomics Tools for Taxonomic Classification* by Simon H. Ye et. al [8]. Note that the $L2$ - distance gives a measure of the overall abundance estimation for the dataset and does not consider the abundance estimation for each individual species.

$$L2 \ = \ \sqrt{\sum_{i=1}^{S}(p_i - t_i)^2}. \tag{4.4}$$

4.2.3 Baseline - Upper Bound Accuracy

To be able to compare the performance of Kraken2 and our neural network models an estimate of the proportion of reads classified correctly on species level is needed. Although it is possible to identify the exact classification of each read using *KrakenTools*[39] it is sufficient to calculate an upper boundary of the classification accuracy which can then be compared to the classification accuracy obtained by the neural network models. Our upper bound accuracy is defined as:

Given a dataset of N reads from C species with equal number of reads originating from each species, the maximal number of reads that can belong to species c_i is $r_{max} = N/C$. We can then define the upper bound of the accuracy as

$$\text{upper bound accuracy} = \frac{1}{N}\sum_{i=1}^{C} r_i, \tag{4.5}$$

where r_i is the largest possible number of reads correctly classified as species c_i. We have defined r_i as

$$r_i = \begin{cases} a_i & if\ a_i < r_{max} \\ r_{max} & if\ a_i > r_{max}, \end{cases} \tag{4.6}$$

where a_i is the number of reads assigned to species c_i by Kraken2.

Since there is a possibility that the abundance for a species is overestimated, i.e. $a_i > r_{max}$, we can know that some of the reads must be classified incorrectly and r_i has therefore been limited to be at most r_{max}. On the other hand, when $a_i < r_{max}$ it is possible that all read are classified correctly although it is not necessarily the case. However, to get an accuracy as high as possible we have assumed that all reads in this case are classified correctly. For clarification, in this metric all reads classified at a higher taxonomic level than species is considered incorrectly classified although it is not completely true as they merely are classified at a higher level.

4.3 Pre-processing

As some of the reference genomes used for simulating the dataset are not complete they have been padded with the character *N*, indicating an unknown base in the genome. Reads containing one or more of the "N-base" are neither realistic from sequencing perspective nor are they containing any real information related to a specific species. To avoid learning that bacterial DNA contains the base N all reads containing one or more N's are removed.

For each of the filtered reads, a k-mer profile was created using the bag-of-words representation presented in section 3.3.1. The size of the k-mer was 7 for *Streptococcus* and *Mycobacterium* datasets, and 6 for the group dataset. As the space complexity for the bag-of-words model is $\mathcal{O}(4^k)$ the k-mer size was chosen in order to balance the need for sufficient many reads per species while capturing unique identifiers in the reads without exceeding the storage capacity of the computer. Further, to give more influence to rare k-mers a TF-IDF transform was applied. To avoid a large difference on the input values to the models the data was scaled to be in the range $[0, 1]$ using min-max scaling.

4.4 Implementation of Neural Networks

For our own classification models we decided to implement three separate neural networks; one each for *Streptococcus* and *Mycobacterium* species, and a combined one for a few *Streptococcucs, Mycobacterium*, and *Enterobacteriaceae* species. Worth noting here is that the classification problem for the neural network models is much simpler compared to the one Kraken2, Bracken, and MetaPhlan2 were faced with. Even though we evaluated datasets containing the same species and generated from the same genomes, the established software had many more possible classes to classify the read as compared to the neural network models. For the neural network models it is assumed that we already have some knowledge about what kind of species the reads belong to, i.e. some rough classification have already been made. With this assumption we can narrow the number of possible species down to a specific species group or genus and therefore hopefully improve the proportion of reads that are correctly classified at the species level.

4.4.1 Training Setup

All three models were implemented as feed forward networks with one hidden layer and the additional hyperparameters for each of the networks are presented in the results, section 5.3. For the hidden layers the *Rectified Linear Unit* (ReLU) was used as activation function. The ReLU function is defined as

$$\sigma(x) \;=\; \begin{cases} x & if\ x\ \geq 0 \\ 0 & if\ x\ < 0 \end{cases} \tag{4.7}$$

Since we are implementing a multi-class classifier the activation function for the output layer was the *softmax* function,

$$h_i \;=\; \sum_j w_{j,i} z_i \;+\; b_i. \tag{4.8}$$

$$\hat{y}(\mathbf{h})_i \;=\; \frac{e^{h_i}}{\sum_{j=1}^{dim(\mathbf{Y})} e^{h_j}}. \tag{4.9}$$

By using softmax the values for the output neurons will be in range $y(\mathbf{h})_i \in [0, 1]$ and can be interpreted as a probability, with the highest value corresponding to the most likely class. As stated in section 3.2, the *cross-entropy* loss function is the standard choice for multi-class classifier and was therefore the natural choice in our case.

To avoid overfitting when training the networks dropout and $L2$ weight regularization was added to the hidden layer as described in section 3.2.1.3. Additionally, variable learning rate was implemented where the learning rate was decreased to half its previous value if it had not been any reduction of the loss during the last 10 epochs and early stopping was applied if no improvement had been seen for the last 20 epochs.

4.4.2 Evaluating Classification Accuracy

To measure the performance of the neural network models we first need to decide what species each read belong to. This is done by simple classifying the read as the species corresponding to the output neuron with the highest value. After doing this we can compare with the correct labels for the reads and measure the proportion of reads that have been correctly classified, giving us a classification accuracy defined as

$$Accuracy \;=\; \frac{1}{N}\sum_{i=1}^{N} I(y_i = argmax(\hat{\mathbf{y}}_\mathbf{i})). \tag{4.10}$$

4.5 Data Exploration

Exploring the data we use to train our neural network models can help us understand why they perform the way they do. A first step to understanding the data is to visualizing it, however due to the high dimensionality of the bag-of-words representation we cannot simply plot our data points. To get around this issue PCA was used to reduce the high dimensional TF-IDF representation into three principal components.

Secondly, we wanted to understand how similar the species within our groups were. In order to do so we used *Basic Local Alignment Search Tool* (BLAST) to pairwise align our sequences. BLAST identifies similar regions in the aligned reads and then compare the nucleotides on each position resulting in a percentage of how identical the sequences are [40].

5

Results

In this section the results of this project will be presented. First we will summarize and analyze the evaluation of Kraken2, Bracken, and MetaPhlan2. We will then present the performance of the neural network models on classifying individual reads. Finally, we will visualize and analyze the datasets used for training our neural network models to understand observed difficulties.

5.1 Evaluation Established Methods

The evaluation of how well Kraken2, Bracken and MetaPhlan2 performs on the datasets described in section 3.1.4 have been summarized in table 5.1 and 5.2. As shown in the table 5.1, at no abundance threshold Kraken2 and Bracken identifies a very large number of species however, when introducing an abundance threshold quickly reduces the number of identified species. From the result reports it can be seen that Kraken2 and Bracken succeeds to find all present species while MetaPhlan2 does not manage to separate those belonging to the *Mycobacterium tuberculosis* complex for the 1928 datasets and the *Streptococcus mitis* group from the strep dataset.

Further, table 5.2 shows that Bracken outperforms both Kraken2 and MetaPhlan2 in regards to abundance estimation of species in the 1928 datasets. On the strep dataset, on the other hand, the average L2 distance for Bracken increases slightly compared to Kraken2. Although the L2-distance is one way to measure the correctness of the total abundance estimation, it does not take into consideration how well the estimation is for each species. Figure 5.1 shows a boxplot of the abundance estimation per species evaluated on the 1928miseq datasets. From the boxplot we can see that Kraken2 overall is quite close in its estimations but the L2-distance is increased due to poor estimations of the *E. coli* and *M. tuberculosis* abundances. Through the size of the boxes in figure 5.1, there is an indication that both Kraken2 and Bracken is more consistent in their abundance estimations compared to MetaPhlan2 across multiple datasets.

Dataset	Software	Number of Species			
		Expected	Median	Range	Median (0.1%)
1928miseq	Kraken2	8	120	5	9
	Bracken	8	120	5	9
	MetaPhlan2	8	7	0	7
1928hiseq	Kraken2	8	68	3	8
	Bracken	8	68	3	8
	MetaPhlan2	8	7	0	7
Streptococcus	Kraken2	4	102	11	12
	Bracken	4	102	11	12
	MetaPhlan2	4	2	0	2

Table 5.1: Number of species identified by the different metagenomics softwares tested in this project. In sub-columns we have, from left to right; the number of expected species in the corresponding dataset, the mean number of species identified across five datasets when no abundance threshold was set, the difference between highest and lowest number of identified species, and the mean number of species identified across five datasets when the abundance threshold was set to 0.1%.

Dataset	Software	Number of Species			Abundance Est.	
		Precision	Recall	F1-score	L2	std
1928miseq	Kraken2	0.889	1	0.94	12.08	0.07
	Bracken	0.889	1	0.94	2.10	0.35
	MetaPhlan2	1	0.88	0.93	18.87	0.67
1928hiseq	Kraken2	1	1	1	14.54	0.03
	Bracken	1	1	1	3.05	0.74
	MetaPhlan2	1	0.88	0.93	21.79	1.31
Streptococcus	Kraken2	0.333	1	0.50	21.62	0.03
	Bracken	0.333	1	0.50	24.87	0.20
	MetaPhlan2	0.5	0.25	0.33	32.62	1.30

Table 5.2: Precision, recall and F1-score calculated for the binary classification problem related to species identification. The metrics are calculated after an abundance threshold of 0.1% first was applied. To the right we have the mean L2-distance and corresponding standard deviation for abundance estimation calculated across five datasets.

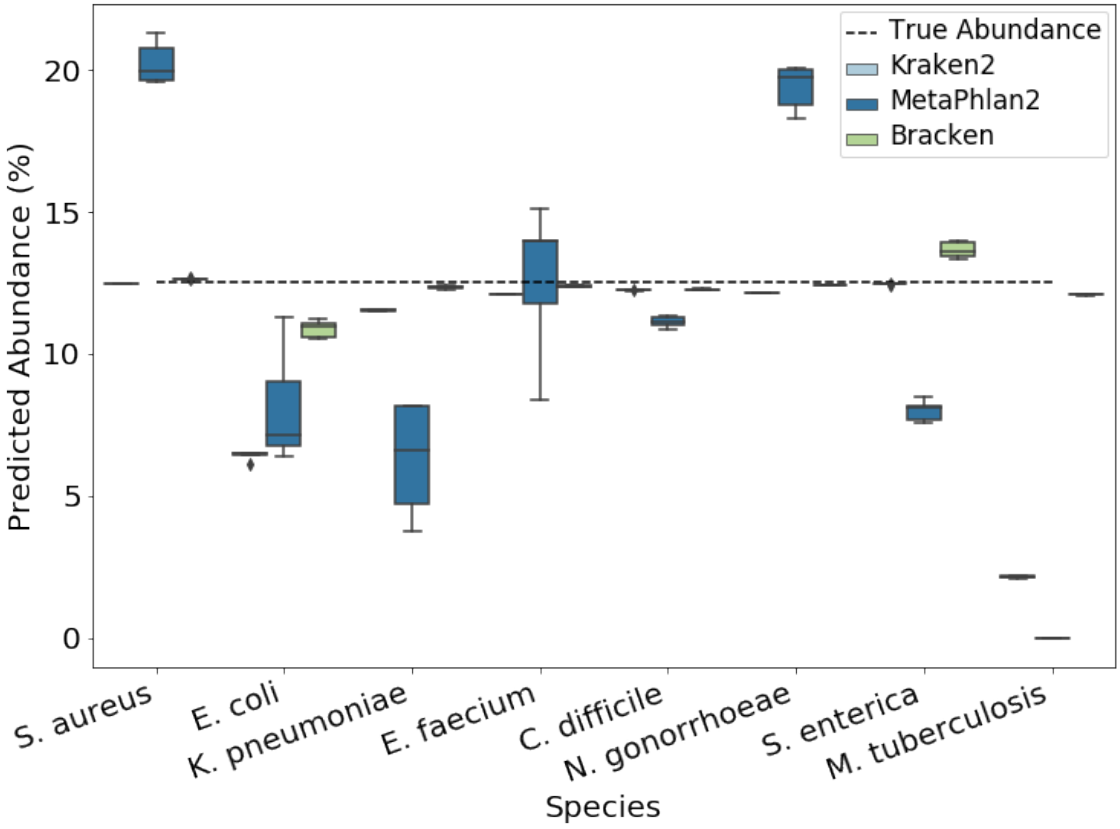


Figure 5.1: Boxplot showing abundance estimation per species for the 1928miseq dataset. The boxes show the quartiles of the predicted abundance and the whiskers are further showing the minimum and maximum values of the predictions from the five separate datasets. That means a larger box indicates larger variance in the abundance prediction for the corresponding species. The expected 12.5% abundance for each species is represented by the dashed, black line.

From the findings presented above, MetaPhlan2 is the tool with the largest limitations; the abundance estimation has the highest variation and is also furthest from the expected value. In addition, MetaPhlan2 does not succeed in separating species belonging to species groups resulting in lower analysis resolution. Although Kraken2 and Bracken succeed in identifying all expected species we can see signs of underperformance on species that has close relatives, in terms of genetic structure. The abundance estimation of *Streptococcus* species is quite high and comparing the values for Kraken2 and Bracken we also notice the overall abundance estimation is not improved by Bracken. The next section presents the results of a more in depth analysis of Kraken2's performance on reads from the *Streptococcus mitis* group and *Mycobacterium tuberculosis* complex.

5.2 Kraken2 on *Streptococcus mitis* group & *Mycobacterium tuberculosis* complex

Testing Kraken2 on datasets with increasing abundance for the *Streptococcus* species figure 5.2 shows that Kraken2 constantly overestimates the abundance of *S. pneumoniae* and underestimates the abundance of *S. mitis* and *S. oralis*. This suggests Kraken2 is slightly biased towards classifying reads as *S. pneumoniae* rather than the commensal *S. mitis* and *S. oralis*. Further, given an expected abundance of 25% per species it was found that the upper boundary of the classification accuracy on species level (proportion of reads classified as the correct species defined as in section 4.2) was 82.29%.

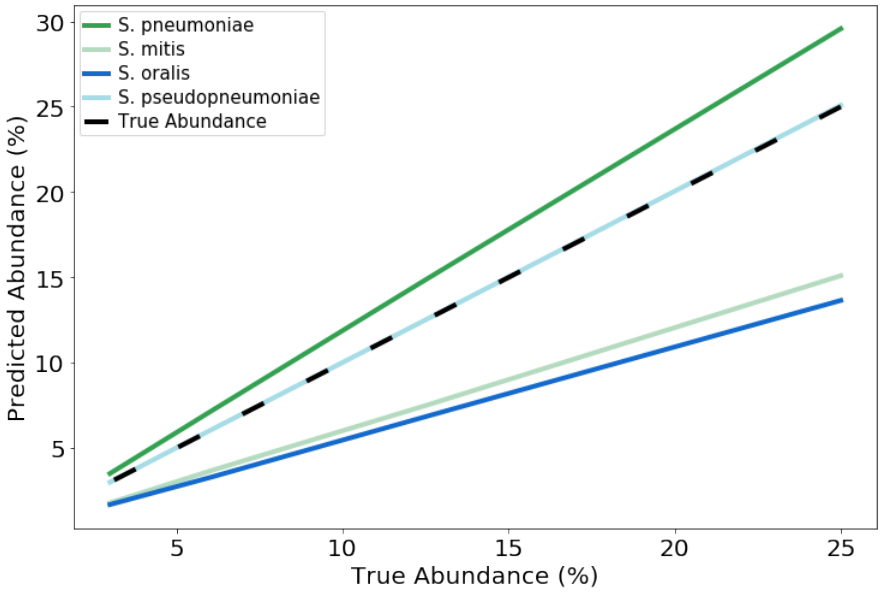


Figure 5.2: Estimated abundance by Kraken2 for species belonging to the *Streptococcus mitis* group. The dashed black line shows the expected abundance for all species.

For species belonging to the *Mycobacterium tuberculosis* complex the Kraken2 report shows that 70.6% of the reads are being classified at genus level instead of the expected species level. This gives an upper bound on the species level accuracy of 28.66%. In addition to this, the report shows Kraken2 considers *M. bovis* and *M. africanum* as subspecies to *M. tuberculosis* rather than species on their own. This seem to be due to NCBI's taxonomy convention and therefore these species have been evaluated on NCBI's *subspecies* level instead of the otherwise considered *species* level.

5.3 Evaluation of Neural Networks

As neural networks has the limitation of only predicting the classes it has been trained on, and there are more than a thousand of bacterial species in NCBI's reference database, it would require a massive training dataset to have a proper representation of each species. Therefore, the suggested application for the neural network model is to be used in a secondary step after an initial classification has been done to, for example, improve the abundance estimation of species groups which the initial method struggle with.

As the data differs for different species groups three separate models were trained; *StrepModel* for *Streptococcus* species, *TubMmodel* for *Mycobacterium* species, and *GroupModel* which is a combined model for *Streptococcus* species, *Mycobacterium* species as well as *Salmonella enterica, Escherichia coli* and *Klebsiella pneumoniae* which are all part of the family *Enterobacteriaceae*. Since the evaluation of Kraken2 was limited to only four species each from the *Streptococcus mitis* group and *Mycobacterium tuberculosis* complex we continued to limit ourselves to the same species as was listed in the section about the simulated datasets, see table 4.1. Moreover, the hyperparameters that was used for training the neural network models are presented in table 5.3.

Hyperparameter	**StrepModel**	**TubModel**	**GroupModel**
Hidden Layers	1	1	1
Neurons Hidden Layer	1200	500	500
Dropout Hidden Layer	0.9	0.0	0.5
Regularization Rate	0.0	0.0	0.0001
Batch Size	1024	1024	1024
Initial Learning Rate	0.001	0.001	0.001
K-mer Size	7	7	6

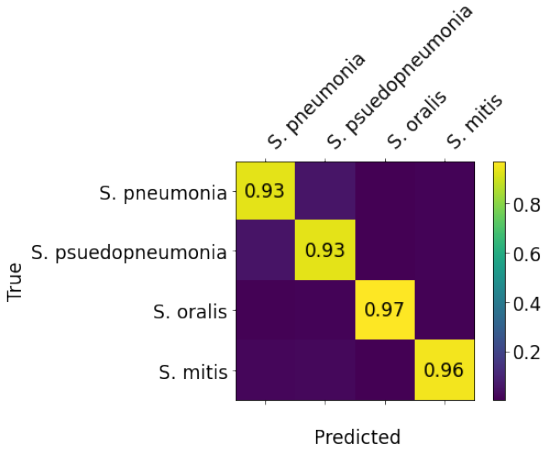
Table 5.3: Hyperparameters used for training neural network classifiers for predicting species identity for each individual read.

Implementation of separate classifiers for the *S. mitis* group and the *M. tuberculosis* complex resulted in test accuracies presented in table 5.4. The combined classifier, *GroupModel*, resulted in a classification accuracy of 66.33%. From the accuracies we can see that our network manages to improve the classification of individual reads on species level compared to what was found using Kraken2 for the investigated *Streptococcus* species. However, the for the *Mycobacterium* species we did not succeed in reaching any significant improvement in classification accuracy compared to Kraken2. Although the performance of the TubModel was quite bad, the results for the StrepModel suggests that neural networks can be a suitable method for improving the classification of reads if we have a limited number of species that we are interested in.

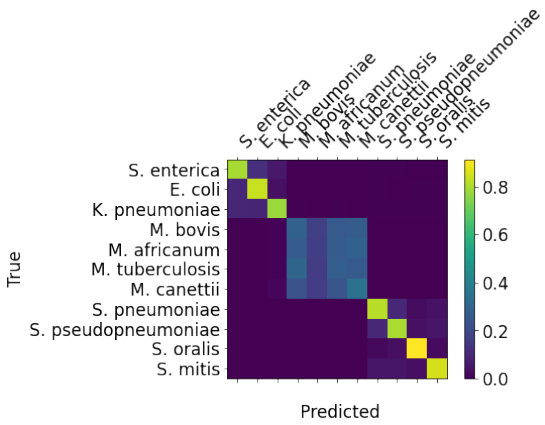
Model/Dataset	NN Classification Accuracy	Kraken2 Accuracy
StrepModel/*Streptococcus*	94.86%	82.29%
TubModel/*Mycobacterium*	30.66%	28.66%
GroupModel	66.33%	-

Table 5.4: Classification accuracy of individual reads obtained for the trained neural network models as well as the estimated accuracy for species level classification of reads obtained by Kraken2. Accuracy refers in both cases to the proportion of reads that was correctly classified at species level.

Figure 5.3a and 5.3b shows the confusion matrix for the *StrepModel* and *GroupModel* respectively. The confusion matrix for the GroupModel clearly tells us that there is no problem for the network to identify what group of species the reads belong to, the misclassifications are instead happening within each species group. In figure 5.3a the confusion matrix for *S. mitis* group shows our network has its largest difficulties in distinguishing *S. pneumonia* from *S. pseudopneumoniae*. As we know from the results in section 5.2, Kraken2 managed to accurately predict the abundance for *S. pseudopneumoniae* and hence a combined method may be possible to utilize the strengths from both methods for an overall improvement.



(a) Confusion matrix given by StrepModel evaluated on a separately generated test set.



(b) Confusion matrix given by GroupModel evaluated on a separately generated test set.

5.4 Exploring the Data

As seen in the results presented in section 5.3 above, our approach of using a feed-forward neural network with a TF-IDF bag-of-words representation of the data showed, in terms of classification accuracy on individual reads, very different results for the *Streptococcus* species and *Mycobacterium* species.

Figure 5.4 shows the first and second principal components for the datasets used to train the *GroupModel* with he data colored according to species. We can see a clear separation of the species groups however, there is no clear separation of the individual species within the groups. This realization helps us understand why there were very few misclassifications in between the groups. The variance explained by the three first principal components for the datasets used for the separate models is presented in table 5.5. The low values for the captured variance for the first principal components verifies that we cannot easily reduce the dimensions without loosing a lot of the variance in the data.

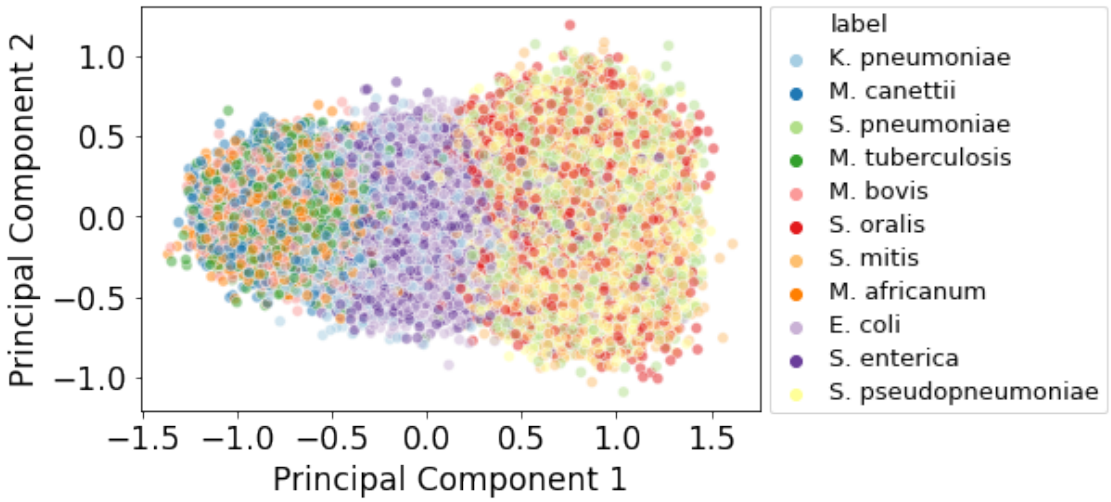


Figure 5.4: First and second principal component for dataset containing *Streptococcus, Mycobacterium* and *Enterobacteriaceae* species. Here a subset of the entire dataset has been used in order to illustrate the overlap of the data for belonging to each species.

Species Group	**PC 1**	**PC 2**	**PC 3**
Streptococcus	0.14	0.13	0.07
Mycobacerium	0.10	0.08	0.07

Table 5.5: Explained variance captured by the three largest principal components identified by PCA for datasets containing *Streptococcus* and *Mycobacterium* species.

Lastly, comparing the pairwise BLAST identity, presented in table 5.6 and table 5.7, between the species belonging to the studied groups we can see that the genome percent identity between the *Mycobacterium* species is above 99%. Comparing that to the maximum identity of 95.69% for *S. pneumonia* and *S. pseudopneumonia* we can realize that separating *Mycobacterium* is a more complex task.

	S. pneumoniae	*S. pseudopneumoniae*	*S. mitis*	*S. oralis*
S. pneumoniae	1.00	0.9569	0.9392	0.9204
S. pseudopneumoniae	0.9569	1.00	0.9462	0.8304
S. mitis	0.9392	0.9462	1.00	0.9484
S. oralis	0.9204	0.8304	0.9484	1.00

Table 5.6: Pairwise BLAST identity for the studied *Streptococcus* species

	M. tuberculosis	*M. africanum*	*M. bovis*	*M. canettii*
M. tuberculosis	1.00	0.9990	0.9988	0.9944
M. africanum	0.9990	1.00	0.9988	0.9953
M. bovis	0.9988	0.9988	1.00	0.9919
M. canettii	0.9944	0.9953	0.9919	1.00

Table 5.7: Pairwise BLAST identity for the studied *Mycobacterium* species

6

Discussion

In this section we will discuss some of the results obtained during this project and also highlight limitations with the chosen methods that needs to be taken into account. We will then round off this thesis with some suggestions on topics for future work on metagenomics.

6.1 Evaluation of Metagenomic Analysis Software

This thesis started out by evaluating three metagenomic analysis software, Kraken2, Bracken and MetaPhlan2, however there are a number of limitations to consider in regards to this evaluation. First, we have only used simulated datasets throughout this project which gives us the the freedom to include the species we are interested in as well as controlling parameters such as abundance distribution. Here, we have restricted ourselves to keep the abundance uniform among the species, this does not give a true representation of metagenomic samples but makes it easier to compare the classification performance between species. The negative side to using only simulated reads is that it is not possible to know if the behaviour of the methods would be the same for ”real” metagenomic samples. For example, the simulated reads are generated from reference genomes that has already been sequenced once. There is always some level of sequencing error which may have transferred over from the reference genomes, resulting in a ”double” error on our reads.

Further, only three tools were evaluated where Bracken also built upon the results of Kraken, and can because of that not be considered as a stand alone tool. Since there are many more tools available there is still plenty left to explore in terms of other software. Also, the evaluation was only conducted on a few specific variables and species where the choice of these most likely affects the outcome of the evaluation. Therefore the results presented here does not give the full picture of the performance of the tools, but can hopefully serve as a first indication that can be further explored once there is a more specific application for it in mind. This is in line with what has been suggested in previous benchmarking studies [9].

In our evaluation we first focused on analyzing the general performance in terms of species identification, abundance estimation, and predictions stability. We found that no single tool outperformed the others on their own as the abundance estimation achieved by Bracken was based on the initial classification by Kraken2. In terms of species identification we found that Kraken2 did a good job identifying all species but also identified a lot of false species. A reason for all these false species comes from errors in the database used to map reads against. In addition to this, some bacteria has genes that have been transferred horizontally and therefore the same gene can be present in multiple bacterial species that not necessarily are close to each other in the taxonomic tree. Since such genes are included in the database, there is a large chance that they will cause misclassifications that will greatly reduce the classification performance.

MetaPhlan2 on the other hand were pretty accurate in its species identification, except that species belonging to a species group were identified as that group, but were significantly worse at abundance estimation. The most interesting find was that all tools struggled more or less with classification of reads from closely related species groups. When estimating the classification accuracy for Kraken2 on the *Streptococcus* and *Mycobacterium* datasets we have made multiple simplifications and assumptions. For example we have only considered reads classified at species level as *correct* and reads at genus level or higher as *incorrect*. Since Kraken2 is a LCA based method, many of the reads belonging to species with similar genetic composition will be classified at a higher level and therefore labeled as incorrect giving a skewed result. Here, maybe another approach which considers the different classification levels would have been better suited to evaluate the quality of the classifications.

Nevertheless, the biggest limitation for the established methods was found to be the species level classification and as the species groups are limited to a smaller number of species, it seemed as this would be the best area to improve using machine learning. We remind here that the classification task for the neural networks is different compared to the one performed by the established metagenomic software. Kraken2, Bracken and MetaPhlan2 have a lot more species that they compare the reads against while the the neural network models only have four species to choose from in the *Streptococcus* and *Mycobacterium* cases and eleven in the GroupModel case. This classification task is evidently easier, but the main goal is also to investigate if we can improve the classification and therefore it makes sense to try simplify the task.

6.2 Implementation and Performance of Neural Network Models

The neural network models we implemented in an attempt to improve the classification of reads from closely related species gave very varying results with a high classification accuracy of almost 95% for the *Streptococcus* model while the *Mycobacterium* model struggled in separating the species and reached only about 30% classification accuracy. When choosing these two groups of species we knew that the *Mycobacterium* species are more similar than the *Streptococcus* species, but we did not expect the difference in the performance to be this big. After realizing that the BLAST identity for the *Mycobacterium* species is above 99% the obtained accuracy is not as surprising. With such a high similarity of the genomes, the short read length, and a sequencing error added on top of that, there is a chance the resolution simply is too low to distinguish the unique features of each species. With that in mind we can understand the difficulty of classifying the species from short reads. One way to possibly be able to improve the performance for this model is to change the representation of the data.

The first thing that should be considered is the use of a larger k-mer size, perhaps we are missing k-mers that are unique between the *Tuberculosis* species because of our smaller k-mer size. Further the classification would maybe be improved if we could take the context of the k-mers into account. Instead of using a bag-of-words representation of the data one can implement *word embeddings* similar to Word2Vec. Word2Vec creates vector representations that are close to each other in the feature space for words that have a similar meaning. An additional approach to capturing the context of the k-mers is to use a *recurrent neural network* architecture instead of the feed forward structure that has been used in this project. The context can be captured with this structure because of the internal feedback loop allowing the model to consider both the input to the model as well as the state of the model in previous time steps.

6.3 Future Work

The field of metagenomics has great potential to revolutionize healthcare and there are many interesting aspects to study. In order for metagenomics to give value in a clinical setting there is a need for a pipeline that accurately detects species for ”real” sequences. As future project it would be interesting to investigate the feasibility for such a pipeline, combining multiple software and possibly even machine learning based models. This thesis has showed that using Bracken on top of Kraken2 to improve abundance estimation had a positive outcome and therefore it would be of great interest to see if other combinations of two or more software can improve the overall taxonomic classification.

When considering the 1928miseq and 1928hiseq datasets we have not taken into consideration if the species naturally occur together. These species were instead chosen because 1928 Diagnostics have analysis pipelines for these species and they are known for causing hospital acquired infections or being resistant to many types of antibiotics. Although the analyzed *Streptococcus* species all inhabits the naso-oral-pharyngeal tract of humans it would in a future study be interesting to focus more on analyzing datasets that are more true to real metagenomic samples. This could for example be to evaluate both simulated and real metagenomic samples corresponding to a blood test from a sepsis patient.

Finally, sequencing techniques that gives long reads of (10,000 - 100,000 bp) have become more accessible and may help improve the classification of species with very similar genomes. If the sequences are longer there is a better chance of capturing unique features of the genome in each individual read. Unfortunately the sequencing quality of long reads are much lower comparing to short reads which may cancel out the positive effects of having longer sequences to analyze.

7

Conclusion

To conclude this thesis we have found that all methods, both established and neural network based, comes with strengths and weaknesses. There is no single approach that on its own is superior to the others and in order for efficient analysis of metagenomic data one should consider a combination of multiple methods. This project also showed that neural networks have the ability to improve certain parts of the classification of metagenomic data, however there may be other more suitable methods that has not been investigated here.

Bibliography

[1] Julian Sutton. *Biology.* Macmillan Education UK, Palgrave, London, 1998. ISBN:9781349152018.

[2] Let's Talk Science. *Radiation Effects on Cells & DNA.* 2020. Available: `https://letstalkscience.ca/educational-resources/backgrounders/radiation-effects-on-cells-dna`. Accessed on: 2020-05-11. [Online].

[3] WHO. *World Health Statistics 2019: Monitoring health for the SDGs.* Technical report, 2019. Available: `https://www.who.int/gho/publications/world_health_statistics/2019/en/` Accessed on: 2020-01-09. [Online].

[4] Marc J Struelens. *The epidemiology of antimicrobial resistance in hospital acquired infections: problems and possible solutions.* BMJ, 317(7159):652–654, 1998. doi:10.1136/bmj.317.7159.652.

[5] Jessica D. Forbes and Natalie C. Knox and Christy-Lynn Peterson and Aleisha R. Reimer. *Highlighting Clinical Metagenomics for Enhanced Diagnostic Decision-making: A Step Towards Wider Implementation.* Computational and Structural Biotechnology Journal, 16:108–120, 2018. doi:10.1016/j.csbj.2018.02.006.

[6] Otávio Guilherme de Almeida and Elaine Martinis. *Bioinformatics tools to assess metagenomic data for applied microbiology.* Applied Microbiology and Biotechnology, 103(1), 01 2019. doi:10.1007/s00253-018-9464-9.

[7] Andrey Mardanov, Vitaly Kadnikov, and Nikolai Ravin. "Metagenomics: A Paradigm Shift in Microbiology" in *Metagenomics*, pages 1–13. 2018. doi:10.1016/B978-0-08-102268-9.00001-X.

[8] Simon H. Ye, Katherine J. Siddle, Daniel J. Park, and Pardis C. Sabeti. *Benchmarking Metagenomics Tools for Taxonomic Classification.* Cell, 178(4):779–794, aug 2019. doi:10.1016/j.cell.2019.07.010.

[9] Michael Peabody, Thea Van Rossum, Raymond lo, and Fiona Brinkman. *Eval-*

uation of shotgun metagenomics sequence classification methods using in silico and in vitro simulated communities. BMC Bioinformatics, 16(362), 11 2015. doi:10.1186/s12859-015-0788-5.

[10] Camilo Mora, Derek Tittensor, Sina Adl, Alastair Simpson, and Boris Worm. *How Many Species Are There on Earth and in the Ocean?* PLoS biology, 9(8):e1001127, aug 2011. doi:10.1371/journal.pbio.1001127.

[11] Catalogue of Life. *Progress*. Available: `https://www.catalogueoflife.org/`. Accessed on: 2020-05-11. [Online].

[12] Tindall, B. J. and Rosselló-Móra, R. and Busse, H.-J. and Ludwig, W. and Kämpfer, P. *Notes on the characterization of prokaryote strains for taxonomic purposes*. International Journal of Systematic and Evolutionary Microbiology, 60(1):249–266, 2010. doi:/10.1099/ijs.0.016949-0.

[13] Gabriela Salvadori, Roger Junges, Donald Morrison, and Fernanda Petersen. *Competence in Streptococcus pneumoniae and Close Commensal Relatives: Mechanisms and Implications*. Frontiers in Cellular and Infection Microbiology, 9:94, 2019. doi:10.3389/fcimb.2019.00094.

[14] U.S. National Library of Medicine. *Help Me Understand Genetics: Cells and DNA*. `https://ghr.nlm.nih.gov/primer/basics/dna`, June 9, 2020. Accessed on: 2020-05-11. [Online].

[15] U.S. National Library of Medicine. *Help Me Understand Genetics: Mutations and Health*. `https://ghr.nlm.nih.gov/primer/mutationsanddisorders/evolution`, June 9, 2020. Accessed on: 2020-05-11. [Online].

[16] Sam Behjati and Patrick S Tarpey. *What is next generation sequencing?* Archives of Disease in Childhood - Education and Practice, 98(6):236–238, 2013. doi:10.1136/archdischild-2013-304340.

[17] J. Besser, H.A. Carleton, P. Gerner-Smidt, R.L. Lindsey, and E. Trees. *Next-generation sequencing technologies and their application to the study and control of bacterial infections*. Clinical Microbiology and Infection, 24(4):335 – 341, 2018. doi:10.1016/j.cmi.2017.10.013.

[18] Prakhar Vijayvargiya, Patricio R. Jeraldo, Matthew J. Thoendel, Kerryl E. Greenwood-Quaintance, Zerelda Esquer Garrigos, M. Rizwan Sohail, Nicholas Chia, Bobbi S. Pritt, and Robin Patel. *Application of metagenomic shotgun sequencing to detect vector-borne pathogens in clinical blood samples*. PLoS ONE, 14(10), 2019. doi:10.1371/journal.pone.0222915.

[19] Sharmila Mande, Mohammed Haque, and Tarini Shankar Ghosh. *Classification of metagenomic sequences: Methods and challenges*. Briefings in bioinformatics,

13(6):669–681, 2012. doi:10.1093/bib/bbs054.

[20] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, Berlin, Heidelberg, 2006.

[21] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[22] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning Data Mining, Inference, and Prediction*. Springer, New York, NY, 2009. doi/10.1007/978-0-387-84858-7.

[23] Mattias Wahde. *Biologically Inspired Optimization Methods: An Introduction*. WIT Press, London, 2008. ISBN:978-1-84564-148-1.

[24] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. arXiv preprint arXiv:1609.04747, 2016.

[25] Xue Ying. *An Overview of Overfitting and its Solutions*. Journal of Physics: Conference Series, 1168(2):022022, 2019. doi:10.1088/1742-6596/1168/2/022022.

[26] Andrew Y. Ng. *Feature Selection, L1 vs. L2 Regularization, and Rotational Invariance*. ICML '04:Proceedings of the Twenty-First International Conference on Machine Learning, page 78, 2004. doi:10.1145/1015330.1015435.

[27] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. *Dropout: A Simple Way to Prevent Neural Networks from Overfitting*. Journal of Machine Learning Research, 15(56):1929–1958, 2014. doi:10.5555/2627435.2670313.

[28] Jason Brownlee. "A Gentle Introduction to the Bag-of-Words Model" in *Deep Learning for Natural Language Processing*. Self-Published, 2017. Available: `https://machinelearningmastery.com/gentle-introduction-bag-words-model/` Accessed on: 2020-05-25. [Online].

[29] Stephen Robertson. *Understanding Inverse Document Frequency: On Theoretical Arguments for IDF*. Journal of Documentation - J DOC, 60(5):503–520, oct 2004. doi:10.1108/00220410410560582.

[30] Gareth James, Daniela Witten, Trevor Hastie, and Robert Tibshirani. *An Introduction to Statistical Learning*. Springer, New York, NY, 2013. doi:10.1007/978-1-4614-7138-7.

[31] Alexa B. R. McIntyre, Rachid Ounit, Ebrahim Afshinnekoo, Robert J. Prill, Elizabeth Henaff, N Vasiliev Alexander, Samuel S Minot, David Danko,

Jonathan Foox, Sofia Ahsanuddin, Scott Tighe, Nur A. Hasan, Poorani Subramanian, Kelly Moffat, Shawn Levy, Stefano Lonardi, Nick Greenfield, Rita R. Colwell, Gail L. Rosen, and Christopher E. Mason. *Comprehensive benchmarking and ensemble approaches for metagenomic classifiers.* Genome Biology, 18(182), 2017. doi:10.1186/s13059-017-1299-7.

[32] Hadrien Gourlé, Oskar Karlsson-Lindsjö, Juliette Hayer, and Erik Bongcam-Rudloff. *Simulating Illumina metagenomic data with InSilicoSeq.* Bioinformatics, 35(3):521–522, feb 2019. doi:10.1093/bioinformatics/bty630.

[33] Inc Illumina. *Genomic Sequencing.* Available: `https://www.illumina.com/documents/products/datasheets/datasheet_genomic_sequence.pdf`, 2010. Accessed on: 2020-05-25. [Online].

[34] Nuala A O'Leary, Mathew W Wright, J Rodney Brister, Stacy Ciufo, Diana Haddad, Rich McVeigh, Bhanu Rajput, Barbara Robbertse, Brian Smith-White, Danso Ako-Adjei, et al. *Reference sequence (RefSeq) database at NCBI: current status, taxonomic expansion, and functional annotation.* Nucleic acids research, 44(D1):D733–D745, 2016. doi:10.1093/nar/gkv1189.

[35] Derrick E. Wood, Jennifer Lu, and Ben Langmead. *Improved metagenomic analysis with Kraken 2.* Genome Biology, 20(1):257, nov 2019. doi:10.1186/s13059-019-1891-0.

[36] Jennifer Lu, Florian P. Breitwieser, Peter Thielen, and Steven L. Salzberg. *Bracken: estimating species abundance in metagenomics data.* PeerJ Computer Science, 3(1):e104, jan 2017. doi:10.7717/peerj-cs.104.

[37] Nicola Segata, Levi Waldron, Annalisa Ballarini, Vagheesh Narasimhan, Olivier Jousson, and Curtis Huttenhower. *Metagenomic microbial community profiling using unique clade-specific marker genes.* Nature Methods, 9(8):811–814, aug 2012. doi:10.1038/nmeth.2066.

[38] Duy Tin Truong, Eric A. Franzosa, Timothy L. Tickle, Matthias Scholz, George Weingart, Edoardo Pasolli, Adrian Tett, Curtis Huttenhower, and Nicola Segata. *MetaPhlAn2 for enhanced metagenomic taxonomic profiling.* Nature Methods, 12(10):902–903, sep 2015. doi:10.1038/nmeth.3589.

[39] Jennifer Lu. *Kraken Tools.* GitHub repository, 2020. Available: `https://github.com/jenniferlu717/KrakenTools/` Accessed on: 2020-05-11. [Online].

[40] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. *Basic local alignment search tool.* Journal of Molecular Biology, 21(3):403–410, 1990. doi:10.1016/S0022-2836(05)80360-2.

[41] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. "pytorch: An imperative style, high-performance deep learning library". In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[42] Travis Oliphant. *NumPy: A guide to NumPy*. USA: Trelgol Publishing, 2006–. Available: `http://www.numpy.org/` Accessed on: 2020-05-25. [Online].

[43] Wes McKinney et al. Data structures for statistical computing in python. In *Proceedings of the 9th Python in Science Conference*, volume 445, pages 51–56. Austin, TX, 2010.

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

[45] J. D. Hunter. *Matplotlib: A 2D graphics environment*. Computing in Science & Engineering, 9(3):90–95, 2007. doi:10.1109/MCSE.2007.55.

[46] Michael Waskom, Olga Botvinnik, Joel Ostblom, Saulius Lukauskas, Paul Hobson, MaozGelbart, David C Gemperline, Tom Augspurger, Yaroslav Halchenko, John B. Cole, Jordi Warmenhoven, Julian de Ruiter, Cameron Pye, Stephan Hoyer, Jake Vanderplas, Santi Villalba, Gero Kunter, Eric Quintero, Pete Bachant, Marcel Martin, Kyle Meyer, Corban Swain, Alistair Miles, Thomas Brunner, Drew O'Kane, Tal Yarkoni, Mike Lee Williams, and Constantine Evans. *mwaskom/seaborn: v0.10.0*, jan 2020. doi:10.5281/zenodo.3629446.

A

Software

In this project a number of software as well as python packages was used for classifying reads and implementing the neural network models. Table A.1 lists the all used software and packages along with where they was used.

Software/Python Package	**Reference**	
Software		
InSilicoSeq	[32]	Generation of datasets
Kraken2	[35]	Species identification and abundance estimation
Bracken	[36]	Species identification and abundance estimation
MetaPhlan2	[37, 38]	Species identification abundance estimation
BLAST	[40]	Alignment of reference genomes
Python Packages		
PyTorch	[41]	Implementation of neural networks
Numpy	[42]	Pre-processing and exploration of data
Pandas	[43]	Pre-processing of data
Scikit - learn	[44]	Pre-processing and exploration of data
Matplotlib	[45]	Visualizations
Seaborn	[46]	Visualizations

Table A.1: Stand alone software and python packages used in during this project.

B

Reference Genomes

Table B.1 lists the accession number for the reference genome for each species that was used to generate datasets. All reference genomes were downloaded for NCBI's RefSeq [34].

Species	**NCBI Accession Number**
Salmonella enterica	NZ_CCNU01000001.1
Klebsiella pneumoniae	NZ_NGTB01000001.1
Escherichia coli	NZ_AKBV01000001.1
Staphylococcus aureus	NZ_CYKD01000001.1
Neisseria gonorrhoeae	NZ_UGRF01000001.1
Enterococcus faecium	NZ_CP012430.1
Clostridioides difficile	NZ_CCRO01000001.1
Mycobacterium tuberculosis	NC_000962.3
Mycobacterium africanum	NZ_CP014617.1
Mycobacterium bovis	NZ_CP012095.1
Mycobacterium canettii	NC_015848.1
Streptococcus pneumonia	NZ_CP018137.1
Streptococcus pseudopneumonia	NC_015875.1
Streptococcus mitis	NZ_CABEHV010000004.1
Streptococcus oralis	NZ_CABEIU010000002.1

Table B.1: List of species used in the datasets and their corresponding NCBI accession numbers.