Bioinformatics
doi.10.1093/bioinformatics/xxxxxx
Advance Access Publication Date: Day Month Year
Manuscript Category

DeePaC: Predicting pathogenic potential of novel DNA with a universal framework for reverse-complement neural networks

Jakub M. Bartoszewicz [1,2,*], Anja Seidel [1,2], Robert Rentzsch [1] and Bernhard Y. Renard [1,*]

[1] Bioinformatics Unit (MF1), Department for Methodology and Research Infrastructure, Robert Koch Institute, 13353 Berlin, Germany
[2] Department of Mathematics and Computer Science, Free University of Berlin, 14195 Berlin, Germany.

* To whom correspondence should be addressed.

Associate Editor: XXXXXXX

Received on XXXXX; revised on XXXXX; accepted on XXXXX

Abstract

Motivation: We expect novel pathogens to arise due to their fast-paced evolution, and new species to be discovered thanks to advances in DNA sequencing and metagenomics. What is more, recent developments in synthetic biology raise concerns that some strains of bacteria could be modified for malicious purposes. Traditional approaches to open-view pathogen detection depend on databases of known organisms, limiting their performance on unknown, unrecognized, and unmapped sequences. In contrast, machine learning methods can infer pathogenic phenotypes from single NGS reads even though the biological context is unavailable. However, modern neural architectures treat DNA as a simple character string and may predict conflicting labels for a given sequence and its reverse-complement. This undesirable property may impact model performance.
Results: We present DeePaC, a Deep Learning Approach to Pathogenicity Classification. It includes a universal, extensible framework for neural architectures ensuring identical predictions for any given DNA sequence and its reverse-complement. We implement reverse-complement convolutional neural networks and LSTMs, which outperform the state-of-the-art methods based on both sequence homology and machine learning. Combining a reverse-complement architecture with integrating the predictions for both mates in a read pair results in cutting the error rate almost in half in comparison to the previous state-of-the-art.
Availability: The code and the models are available at: https://gitlab.com/rki_bioinformatics/DeePaC
Contact: BartoszewiczJ@rki.de, RenardB@rki.de

1 Introduction

1.1 Motivation

Bacterial pathogens evolve quickly and our globalized world enables fast transmission of causative agents over great distances. Virulence genes may be easily exchanged between many bacterial species, leading to emergence of new biological threats. For instance in 2011, a strain of *Escherichia coli* that acquired Shiga-toxin producing genes caused a major outbreak leading to 53 deaths (Frank *et al.*, 2011).

Since DNA sequencing has become the state-of-the art in open-view pathogen detection (Lecuit and Eloit, 2014; Calistri and Palù, 2015), new pipelines and algorithms are needed to efficiently and accurately process the wealth of data resulting from every run. The number of sequences deposited in public databases grows exponentially, and a new challenge is to design computational tools for dealing with big sets of very divergent sequences (Piro *et al.*, 2018). What is more, Locey and Lennon (2016) predicted that up to a trillion microbial species may be inhabiting the planet. If this debated estimate (Willis, 2016) is correct, about 99.999% of the microbial biodiversity remains to be discovered. Both still unknown and newly emerging organisms may pose a public health threat. The risks are difficult to anticipate, but quick assessment is mandatory.

Recent developments in the field of synthetic biology have raised concerns about the possibility of creating new biological threats in the lab, either by accident or for malicious purposes. A report by the National Academies of Sciences, Engineering, and Medicine (2018) argues that genetic modification of existing bacteria to make them more dangerous

© The Author 2019.

is an issue of the highest concern. New methods for sequence-based classification of potential pathogens must be developed to safeguard future biosecurity and biosafety alike. This is also supported by previous work by the National Research Council (2010). List-based approaches, which can only detect previously known organisms by comparing sequences to a reference database, are insufficient. This is especially important for shorter sequences, like next-generation sequencing (NGS) reads or synthetic oligonucleotides. The latter are often not screened before synthesis due to high computational cost and low accuracy of the predictions (Carter and Friedman, 2015; National Academies of Sciences, Engineering, and Medicine, 2018).

Assessing and mitigating risks based on the DNA sequence alone should involve computational methods able to recognize relevant patterns and generate predictions for novel sequences. Therefore, machine learning based approaches are a promising alternative to the traditional sequence analysis tools. For example, Nielsen and Voigt (2018) used deep convolutional networks to predict the lab of origin of the plasmids available in the Addgene repository. Their method could help track a biological threat back to its source in case of a malicious attack or accidental release. Deneke *et al.* (2017) presented a random forest approach for predicting whether an Illumina read originates from a pathogenic or a non-pathogenic bacterium and showed that it generalizes to novel, previously unseen species. They introduce the concept of a *pathogenic potential* to differentiate between predicted probabilities of a given phenotype and true pathogenicity, which can only be realized in the biological context of a full genome and in interaction with the host.

1.2 Computational tools for pathogen detection

1.2.1 Taxonomy-dependent

Read-based pathogen detection methods may be roughly divided in two categories. Taxonomy-dependent approaches directly rely on lists and databases of known pathogens, aiming at assigning sequences to taxonomic categories. Read mappers, for example BWA (Li and Durbin, 2009) and Bowtie2 (Langmead and Salzberg, 2012) fall into this category. Live mapping approaches, such as HiLive and HiLive2 (Lindner *et al.*, 2017; Loka *et al.*, 2018), are even able to map the reads in real time, as the sequencer is running. They offer precise intermediate results and a drastic reduction of the total analysis time.

In general, read mappers specialize in computationally efficient alignment of NGS reads to reference genomes with high specificity. For this reason, they are routinely used for detection of known pathogens, but do not perform well when a sample contains organisms absent from the reference index. Specialized pipelines (Hong *et al.*, 2014; Andrusch *et al.*, 2018) use read mappers in conjunction with additional filtering steps for accurate diagnostics for clinical samples. BLAST (Altschul *et al.*, 1990) offers much more sensitive alignment, appropriate for between-species comparisons, but at a price of much lower throughput.

Metagenomic profiling tools may also be used as taxonomy-dependent pathogen detection methods. Kraken (Wood and Salzberg, 2014) builds a database of unique 31-mers to assign reads to their corresponding taxonomic units. Ambiguities are resolved by returning the lowest common ancestor. MetaPhlAn2 (Truong *et al.*, 2015) uses around 1 million clade-specific marker genes to detect sequences matching its reference genomes. MicrobeGPS (Lindner and Renard, 2015) identifies potentially unknown organisms present in a metagenomic sample and estimates genomic distances between them and known references. NBC (Rosen *et al.*, 2008, 2011), a naïve Bayes classifier, is a machine learning based method trained to recognize taxa based on their k-mer frequency profiles.

1.2.2 Taxonomy-agnostic

Taxonomy-agnostic methods strive to predict phenotypes directly from DNA sequences, without performing any taxonomic assignment. They are not entirely taxonomy-independent, as they must be trained using available references. This may lead to bias with regard to the over- and underrepresented taxa in a training set. Therefore, the goal of the taxonomy-agnostic approaches is to minimize that bias and offer relatively accurate predictions even for novel and divergent sequences. In contrast, the taxonomy-dependent methods are entirely based on the correspondence between a phenotype and a taxonomic classification.

As NBC allows constructing a custom reference database, it may be used in a taxonomy-agnostic manner. However, it is outperformed by PaPrBaG (Deneke *et al.*, 2017), a random forest approach based on a wide range of k-mer and peptide-based features. Although PaPrBaG is a read-based method, it can be also used to predict a phenotype from a whole genome. Any long sequence may be fragmented into simulated reads or read-length subsequences. A mean over all the predictions constitutes the final prediction by majority vote. Although this approach is limited to detecting local effects and cannot use a wider genomic context, it may be useful in practice. Barash *et al.* (2018) used PaPrBaG and their original tool BacPaCS to predict labels for novel genomes recently deposited in the PATRIC database (Wattam *et al.*, 2017). We note that this can also be a possible application of read-based pathogenic potential prediction.

1.3 Deep learning for DNA sequences

Deep learning has been successfully used on genomic data to detect genome accessibility (Kelley *et al.*, 2016) or transcription factor binding sites and disease-associated variants (Alipanahi *et al.*, 2015; Zhou and Troyanskaya, 2015; Zeng *et al.*, 2016; Quang and Xie, 2016; Greenside *et al.*, 2018). Budach and Marsico (2018) implemented convolutional neural networks (CNNs) and long short-term memory networks (LSTMs) in a recently published package, pysster. They integrate RNA sequences and their secondary-structure to predict A-to-I editing and protein binding sites. The package may be used also for DNA and protein sequences. In a different application, Poplin *et al.* (2018) use CNNs to call SNPs and small variants from NGS reads already aligned to a reference genome.

1.4 Reverse-complement neural networks

Although popular deep learning architectures such as CNNs and LSTMs may be successfully applied to DNA sequencing data, they fail to explicitly model one of the most striking relations between DNA sequences, the reverse-complementarity. This is especially important in the context of next-generation sequencing, where an isolated read may originate from either of the two DNA strands. Any motif or sequence pattern should be detected, regardless of its orientation. Shrikumar *et al.* (2017) proposed to use reverse-complement CNNs (RC-CNNs) to ensure identical predictions for both strands. They developed reverse-complement versions of convolutional layers, batch-normalization layers, a weighted-sum layer relevant to their particular application (transcription factor binding site prediction), and a dense layer integrating the forward and reverse-complement representations. However, their method is based on the reverse-complement convolutional filters and thus only applicable to convolutional networks. An analogous approach was independently proposed by Onimaru *et al.* (2018).

1.5 Pathogenic potential prediction with RC-networks

In this work, we present DeePaC, a Deep Learning Approach to Pathogenicity Classification. We focus on the scenario of pathogen detection from next-generation sequencing data. However, the method presented here can be in principle used also for other sequences of similar length. We also show that the traditional deep learning architectures suffer from major problems when trained on DNA sequences, yielding contradictory predictions for a given sequence and its reverse-complement.

In the case of isolated, short reads or oligonucleotides, there is no biological context available and those differences cannot be justified. In a risk-sensitive setting, this is a large vulnerability. To solve this problem, we propose a universal framework for reverse-complement neural architectures, guaranteeing identical predictions for both DNA strands.

2 Methods

2.1 Data preprocessing

2.1.1 IMG dataset

Similarly to Deneke *et al.* (2017), we accessed the IMG/M database (Chen *et al.*, 2019) on April 17, 2018, and followed the procedure described to identify pathogenic and non-pathogenic bacteria with a human host. Briefly, we filtered the database searching for keywords resulting in unambiguous assignment of a given strain to one of the classes. Previous research successfully used both permanent draft and finished genomes in a similar setting. We decided to include draft genomes in our dataset, too, as their quality is similar to the quality of permanent drafts. For seven species of well-known pathogens (*Campylobacter jejuni*, *Clostridioides difficile*, *Clostridium botulinum*, *Francisella tularensis*, *Listeria monocytogenes*, *Staphylococcus aureus*, and *Staphylococcus epidermidis*), we found between one and two non-pathogenic strains and multiple pathogenic strains; we removed the non-pathogenic strains from further analysis. For *E. coli*, we found multiple strains of both labels. We decided that a study focusing on pathogens should be rather more than less sensitive to this particularly diverse species, especially since some strains may lead to dangerous outbreaks, like the EHEC epidemic in Germany in 2011 (Frank *et al.*, 2011). Therefore, we also removed the non-pathogenic strains of *E. coli* from the final dataset. The procedure yielded a collection of 2,878 strains (2,796 pathogens and 82 non-pathogens) described by data including species names, NCBI Bioproject IDs, and class labels. We then linked the Bioproject IDs with GenBank assembly accession numbers found in the GenBank assembly summary file downloaded on April 17, 2018 as well. We downloaded the assemblies and selected one strain per species at random. This was necessary to avoid skewing the model's performance towards species with many sequenced strains available.

2.1.2 Species-level validation

This resulted in a list of 446 species (389 pathogens and 57 non-pathogens), with a genome of a single strain representing each species. We assigned 80% of the species to the training set, and 10% to the validation and test sets each, keeping the ratios between pathogens and non-pathogens the same between all sets. Using the Mason read simulator with the Illumina error model (Holtgrewe, 2010) and a custom R script, we simulated 10 million single-end Illumina reads per class from the training set and 1.25 million reads per class from the validation set, thus balancing the number of data points in each of the classes. We simulated 1.25 million paired reads per class from the test set, allowing us to evaluate the classification performance in both the single-end and the paired-end setting. This amounts to 80%, 10%, and 10% of all reads assigned to each of the sets, respectively. Read length was set to 250 bases in all cases, representing what is routinely available using an Illumina MiSeq device; for read pairs we used a mean fragment length of 600 with a standard deviation of 60.

2.1.3 Class weighting

While the procedure described above results in a balanced training dataset, the mean coverage of pathogen and non-pathogen genomes is drastically different. To explore an alternative way of solving the class imbalance problem, we also simulated an imbalanced training set, where the total of 20 million training reads was simulated with equal mean coverage from all the training genomes, regardless of their labels. This dataset was then used to train ten networks using a class-weighted loss function (see section 2.3.4 for details). PaPrBaG, constiuting the state-of-the art in machine learning based pathogenicity prediction, does not support error weighting (see section 2.4.1). For BLAST, a method based on sequence homology, this distinction does not apply at all, as its reference database is constructed over whole genomes (see section 2.4.2). However, this does not influence our final results; the class-weighted networks were outperformed by those trained on a balanced set and were not selected as our final models (see section 3.2.1).

2.1.4 Read-level validation

Note that in our primary dataset we placed separate species in training and validation sets to explicitly force our classifiers to predict pathogenic potentials of reads originating from *novel* species. A simpler approach would comprise simulating the reads first, and then assigning them to the training and validation sets. However, this would lead to solving a similar, but different biological problem (classifying reads originating from *known* organisms). To explicitly test those assumptions, we generated a version of the training and validation set where reads from the same species occur in both training and validation sets. In this setting, the test set remained the same as above, so it would be possible to compare the effects of read-level and species-level validation.

2.1.5 Temporal hold-out data

We accessed the IMG/M database again on January 17, 2019 and preprocessed it as described above. We identified three new species passing the filters applied. All of them were pathogens belonging to the *Pantoea* genus, which is absent from our original dataset. We downloaded the genomes and simulated paired reads as described above. To keep the mean coverage at a similar level to the coverage of the original test set, we simulated 100,000 reads in total.

2.1.6 Real data case study

To test the performance of our classifier on data from a real sequencing run, we searched the SRA database for sequencing runs performed for a well-known pathogenic species *Staphylococcus aureus*. This species was not present in the training set (but had rather randomly been assigned to the validation set). We considered runs performed on the Illumina platform with paired-end reads of length 250. Note that in case of real sequencing reads, the true read length may vary, and some of the reads were significantly shorter. We accessed the archive SRX4814864, originating from a lesion swab from an Italian paediatric hospital (Manara *et al.*, 2018). We downloaded the data from the corresponding run SRR7983698 as unclipped FASTA files.

2.1.7 BacPaCS dataset

Finally, we downloaded all the data used by Barash *et al.* (2018) in the assessment of their BacPaCS method. We accessed the PATRIC database (Wattam *et al.*, 2017) by the IDs provided by Barash *et al.* (2018), copying their training and test sets for direct comparability. The original training set consisted of 17,811 genomes of pathogens and 3,274 genomes of non-pathogens, while the test set contained 60 pathogens and 40 non-pathogens. Importantly, those genomes did not represent unique species and the number of strains per species could vary greatly. However, following Barash *et al.* (2018), we treated each strain as a separate entity and used all of them in the analysis. We randomly reassigned 10% of the original training genomes to the validation set, so our BacPaCS training set comprised of 90% of the original. The test set was left unchanged. The read simulation was then performed exactly the same as described in the section 2.1.2.

2.2 Universal reverse-complement framework

2.2.1 DNA encoding and reverse-complementarity

We use distributed orthographic representations of DNA sequences, a method based on one-hot encoding of every nucleotide in the sequence. Namely, a sequence is converted into a 2D binary tensor, where one dimension represents the position of a given nucleotide in the sequence, and the other represents the one-letter nucleotide code as a one-hot encoded vector. Positions of the nucleotide vector correspond to adenine, cytosine, guanine and thymine, respectively. For example, adenine is represented by the vector (1,0,0,0) and thymine by (0,0,0,1). Unknown nucleotides may be encoded as (0,0,0,0). Note that reversing the sequence tensor along the nucleotide axis results in obtaining a complement sequence and flipping it along both axes results in the reverse-complement. Shrikumar *et al.* (2017) have used this fact designing the reverse-complement CNNs (RC-CNNs). For each convolutional filter in the network, they apply both the original weight matrix, and its reverse-complement analogue obtained by reversing both axes. This results in obtaining two convolutional filters with shared parameters, which are guaranteed to detect reverse-complement DNA motifs. The output of such a filter is structured so that the channels i and $n-1-i$ (where n is the total number of channels) represent the same motif, but in opposite orientations. Therefore, deeper layers of the network may use the information contained on both the forward, and the reverse-complement strand. After the convolutional layers, a variant of the dense layer integrates both representations by summing all activations while applying identical weights to each pair of channels i and $n-1-i$. Following dense layers may be used as usual. While the method works perfectly for the convolutional networks, it is not directly applicable to other architectures.

2.2.2 Branched RC-networks

We propose to solve this problem by reverse-complementing the input, instead of the weight matrices of the individual units in a layer. We design networks with two separate branches processing each of the input orientations respectively. Each of the branches consists of identical layers and all the parameters are shared between each pair of layers. We propose two variants of this architecture. In the *full* RC-networks, input to the deeper layers consists of concatenated outputs of both the forward and reverse-complement versions of the previous layer. The output of the RC layer is flipped before concatenating, so that the channels i and $n-1-i$ in the final tensor correspond to the same feature on opposing strands. Note that in this case, a *full* RC-CNN is equivalent to the RC-CNNs proposed by Shrikumar *et al.* (2017). In the second variant, dubbed a *siamese* RC-architecture, each of the branches functions separately before the merging layer. That means that the input to a deeper layer is just the output of the previous layer in a branch. We test both variants using CNNs, bidirectional LSTMs and hybrid networks with both convolutional and LSTM layers, but they are in principle compatible with any other neural architecture.

2.2.3 Representation merging

The final layer merges representations of both strands. While in the original RC-CNNs, this is done by summing the activations of corresponding channels, we note that this can be done by any element-wise operation on the 1D output tensors. Apart from the Shrikumar-style summation, we consider two alternative merging functions. The first is the *max* function, implementing the Gödel *t*-conorm, corresponding to the OR operation in Gödel fuzzy logic. Even though the activations are not restricted to the interval [0,1], high output values can be interpreted as finding a motif on either of the two strands. The second function is the Hadamard product (element-wise multiplication), or the product *t*-norm corresponding to the AND operation in the product fuzzy logic. Here, high values may be understood as finding a motif on both strands at the same time.

2.2.4 Species-level and paired reads predictions

One of the major challenges of pathogenic potential prediction from single reads is the lack of biological context. However, if all the reads in a sample originate from the exactly same organism, we can predict the pathogenic potential of that organism by a majority vote. In the context of probabilistic estimates of the class label (returned by both PaPrBaG and our neural networks), we can implement that as a simple mean over predictions for all the individual reads. For BLAST, we can just assign the label predicted for the majority of reads.

Building upon this idea, we can boost read-based performance if we consider read pairs, assumed to originate from the same organism even in metagenomic samples. To this end, we average predictions for the corresponding pairs in our test set. The classifiers may still predict pathogenic potentials for isolated sequences if so desired. We can integrate binary predictions (e.g. returned by BLAST, see section 2.4.2), taking into account the missing and conflicting predictions for some of the reads. We treat missing predictions as undefined values and implement the *accept anything* operator of ternary logic. It returns a positive label if and only if one of the input values is positive, and the other is not negative. Conversely, it returns a negative label if and only if one of the input values is negative, and the other is not positive. In the remaining cases (i.e, in case of conflicting input values or if both inputs are undefined) the result is undefined.

2.2.5 Implementation details

We implemented the architectures and all steps needed for preprocessing the data, training, prediction and evaluation of the resulting models. We used Keras 2.2.4 (Chollet and others, 2015) with TensorFlow 1.12 (Martín Abadi *et al.*, 2015) as backend for the deep learning parts of the code, as well as Biopython 1.72 for efficient fasta file preprocessing. The branched RC-architecture is easy to parallelize. We use model parallelism when training the RC-LSTM networks - each of the branches may process the data on a separate GPU, while the third GPU stores the model weights and performs the merging and dense layer operations. We use a fast CuDNNLSTM implementation by default. For RC-CNNs, we also support data parallelism, or passing multiple batches of data through the network at the same time, one on each of the available GPUs. The source code is available at https://gitlab.com/rki_bioinformatics/DeePaC.

2.3 Training and tuning

2.3.1 Hardware

Each of the networks was trained on between one and four GPUs (GTX 980, GTX 1080 Ti, or RTX 2080 Ti) depending on hardware availability at a given time point. CUDA 9.0 was used on machines equipped with the GTX-line cards, but the RTX 2080 Ti card required CUDA version 10.0 and compiling TensorFlow from source for compatibility.

2.3.2 Hyperparameter tuning: RC-LSTM

For hyperparameter tuning, we tested a total of 243 different architectures. All CNNs and LSTMs were initialized with He (He *et al.*, 2015) weight initialization and trained with the Adam optimizer (Kingma and Ba, 2014) using the default parameters and a batch size of 512. We finished the training after a maximum of 15 epochs or earlier if the validation accuracy did not improve for 10 consecutive epochs. We used dropout regularization (Srivastava *et al.*, 2014) with a dropout rate of 0.5 after all recurrent and dense layers, and input dropout (interpreted as setting a random fraction of nucleotides to Ns) in most of the architectures. For the bidirectional RC-LSTMs, we tested the following parameter combinations for both the *full* and the *siamese* variant:

- sum merge, 1 layer, 128-512 units, input dropout rate: 0, 0.1-0.3

- max merge, 1 layer, 384 units, input dropout rate: 0.1-0.3
- product merge, 1 layer, 384 units, input dropout rate: 0.1-0.3
- sum merge, 2 layers, 384 units, input dropout rate: 0.1-0.3

The input dropout rate was varied with step of 0.05 unless stated otherwise, and the number of units was adjusted with a step of 128. In addition, we trained two traditional LSTMs without RC parameter sharing, with 256-384 units and an input dropout of 0.2.

2.3.3 Hyperparameter tuning: RC-CNN

For the RC-CNN architectures, we used 64-512 units in the convolutional layers and 64-256 units in the dense layers. Six unit number combinations were generated by alternate doubling of one of those values at each step, starting with the number of convolutional units. We tested the following networks with sum merge, filter size of 15 and input dropout rates of 0 and 0.2-0.3 unless stated otherwise:

- *full*-RC, max pooling, 1 conv. and 1 dense layers
- *full*-RC, average pooling, 1 conv. and 1 dense layers
- *full*-RC, max pooling, 2 conv. and 2 dense layers
- *full*- and *siam*-RC, average pooling, 2 conv. and 2 dense layers
- *full*-RC, average pooling, 3 conv. and 3 dense layers, input dropout rate 0.2-0.25

We also applied batch normalization (Ioffe and Szegedy, 2015) to the RC-CNNs with 2 convolutional and 2 dense layers of 512 and 256 units (hereafter: 2x2-XL), and tested changing the filter size to 7 or 11 by training *full* RC-CNNs with the same layout. In addition, we trained two traditional CNNs without RC parameter sharing. One was a 2x2-XL architecture with the input drouput of 0.25, and the other was a 1x1 max pooling network with 256 convolutional and 128 dense units. Note that they are equivalent to the networks that can be implemented with the pysster package (Budach and Marsico, 2018). Finally, we evaluated hybrid *full* and *siamese* RC-networks with a convolutional layer of 512 units and filter size of 7, 11 or 15, followed by a recurrent layer of 384 units, trained with the input dropout of 0.2. Those networks were trained for just 10 epochs due to high computational cost.

2.3.4 Loss function and training sets

We use binary cross-entropy for all training runs. We tested the effect of adding an L_2 regularization term while training both the *full* and *siamese* RC-LSTMs with sum merge, 1 layer of 384 units and an input dropout rate of 0.2. We used regularization rates of 10^{-2}, 10^{-3} and 10^{-5}. Next, we tested using an imbalanced training set and weighting the errors by the inverse of the relative class frequency. To this end, we trained *full* and *siamese* RC-LSTMs with sum merge, 1 layer of 384 units and input dropout rates between 0.1 and 0.3. Finally, we selected the best CNN and the best LSTM model, and prepared a simple ensemble classifier by averaging the predictions of those two models.

2.3.5 Read-level validation

Generalization from one set of reads to another set of reads should be relatively easy if both sets originated from the same species. However, we expected that even very high read-level validation accuracy would not translate into high test accuracy (see section 2.1.4). After the primary tuning procedure described in the previous sections, we selected an architecture obtaining the highest *training* accuracy (*full* RC-CNN, 2x2-XL with batch normalization, no input dropout), as we assumed little overfitting would be seen at validation stage in this setting. We trained the network using read-level validation and evaluated it on our primary test set containing reads from species absent in both validation and training sets.

2.4 Benchmarking

2.4.1 PaPrBaG

To benchmark our method against the state-of-the art in pathogenic potential prediction, we trained PaPrBaG random forests on our training set, and evaluated them on our test set. We used two different feature settings for PaPrBaG. The original authors extracted more than 900 features from each of the reads, including k-mer frequencies and a selection of amino-acid and peptide features obtained by a six-frame translation of a read. However, they show that the translation-based features contribute relatively little to the final classification decision. Therefore, we decided to use both the original feature set, and a reduced set comprised of the DNA features only.

2.4.2 BLAST

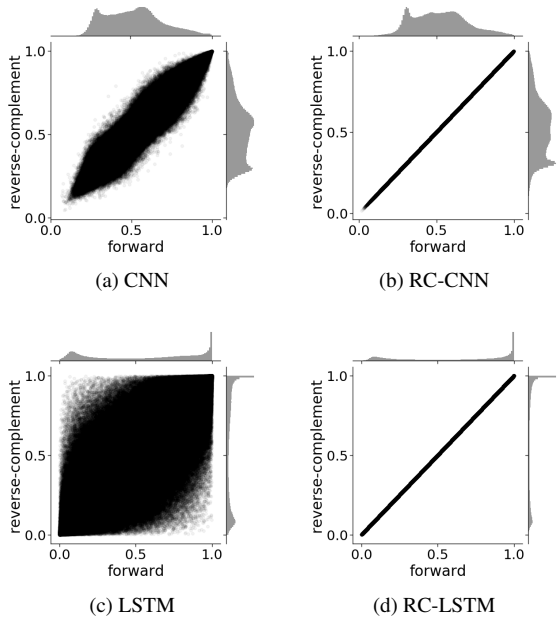
BLAST was shown by Deneke *et al.* (2017) to achieve the best read-by-read performance among alternatives to machine learning approaches to pathogenic potential prediction. It outperformed mapping-based Bowtie2 and its subsequent refinement Pathoscope2, as well as two different variants of k-mer based Kraken (Langmead and Salzberg, 2012; Hong *et al.*, 2014; Wood and Salzberg, 2014). All three failed to classify most of the reads. What is more, BLAST achieved better classification results than a naïve Bayes classifier (NBC) with a word length of 15 (Rosen *et al.*, 2011).

For each test read, we perform a search against the database containing all the training genomes and take a label of the best hit for each of the test reads as a predicted label for that read. We also test a variant of the same approach, but using all the available strains of the training species to build the database.

2.4.3 BacPaCS

Finally, we compared our approach to BacPaCS, Pathogenfinder, and PaPrBaG using the original BacPaCS test dataset (Barash *et al.*, 2018). Without any further tuning, we selected one CNN and one LSTM architecture which worked best on our data. We retrained them both using the BacPaCS training data (see section 2.1.7). Since the BacPaCS dataset treats strains, not species, as the primary entities (and is used by the authors to predict labels for new strains of species *present* in the training set), the classifier designed by Barash *et al.* (2018) can be treated as specialized in predicting pathogenic potentials for *known* species. Therefore, we assumed that the architecture used for read-based validation would yield high performance, and retrained it on BacPaCS data as well (see section 2.1.4). We evaluated the networks in a single-species sample setting and compared the results to the originally presented performance metrics for BacPaCS and Pathogenfinder.

Assessing PaPrBaG, Barash *et al.* (2018) used the random forests trained on the original PaPrBaG dataset. This may lead to inaccurate estimates of the classification error, as the labels they mined from the PATRIC database differ from the labels that the original PaPrBaG forests used. This problem is exacerbated by the imbalance between the number of strains per species in the BacPaCS test dataset. For example, strains of *Acetinobacter baumanii* alone constitute 30% of the non-pathogens in the set. *Fusobacterium periodonticum* amounts to another 10%. Importantly, both of those two species were treated as pathogens in the original PaPrBaG dataset, but are assigned a non-pathogenic label based on the PATRIC metadata. One should therefore expect that the original PaPrBaG forests will predict wrong labels for a significant fraction of the test set if not retrained using the labels extracted by Barash *et al.* (2018) for their training set. Therefore, we retrained PaPrBaG on the original BacPaCS dataset for an accurate comparison to our networks.



(a) CNN (b) RC-CNN

(c) LSTM (d) RC-LSTM

Fig. 1: Prediction distributions for traditional (1a, 1c) and reverse-complement networks (1b, 1d). Predictions for each of the reads (x-axis) are plotted against the predictions for reverse-complements of the same reads (y-axis). The RC-architectures guarantee identical predictions for both strands, which can be strikingly divergent for the traditional LSTM.

3 Results

3.1 Reverse-complementarity constraint

Shrikumar *et al.* (2017) have already shown that standard neural networks do not guarantee identical predictions for a given DNA sequence and its reverse-complement, even if the training dataset is augmented with reverse-complements of all the original sequences. To visualize how the RC neural networks solve this problem, we plot predictions for each read in the test set against predictions for its reverse-complement (Figure 1). Note that the reads are simulated from either of the strands at random. Even though the predicted pathogenic potentials are highly correlated ($\rho > 0.999$, $p < 10^{-6}$) and identically distributed ($p > 0.999$, two-sample KS test) for both orientations, they are not equal. The differences are striking especially for the LSTM model. For the RC neural networks, the predictions are guaranteed to be exactly the same. Interestingly, the prediction distributions are very different for the CNN and LSTM models. We can only speculate that this could be related to the specific properties of the architectures – while the CNNs assume translational invariance and hence focus on localized sequence motifs, the LSTMs are able to also detect long-range relationships between more distant fragments of a read.

3.2 Pathogenic potential prediction from NGS reads

3.2.1 Single reads

We present the results of evaluation on single NGS reads in Table 1. The values given are calculated over the complete test set, treating each read as a separate entity. Precisely, we can calculate the performance measures for the sets of "left" and "right" mates separately and then compute the mean for any given measure. Results obtained for the "left" and "right" half of the set did not differ from those presented in Table 1 by more than 0.001.

As already shown by Deneke *et al.* (2017), BLAST fails to classify some of the reads. To compare its performance to the machine learning approaches, we define accuracy as the ratio of correct predictions to the number of all data points in a set. Therefore, missing predictions are counted as false positives or false negatives depending on the ground truth label. True positive rate (recall) and true negative rate (specificity) are defined in an analogous manner. The fraction of classified reads is presented in the last column.

The neural networks clearly outperform both BLAST and PaPrBaG in terms of prediction accuracy. The selected RC-CNN model consists of 2 convolutional and 2 dense layers with 512 and 256 units respectively; it was trained with an input dropout rate of 0.25 and without batch-normalization. The RC-LSTM has one layer of 384 units, and was trained with input dropout of 0.2. During validation, the networks trained on an imbalanced dataset with a class-weighted loss function performed worse than those trained on a balanced one. The same applied to the traditional deep learning architectures when compared with their reverse-complement counterparts (data not shown). However, the differences in validation accuracy were small (below 0.01).

The RC-CNN model achieves the highest positive predictive value (precision), and the RC-LSTM is the most sensitive. The RC-CNN+LSTM ensemble model (see section 2.3.4) successfully aims at a trade-off between those two performance measures. As expected, the model trained with read-level validation (see section 2.1.4) performs much worse, even though it achieved an impressive validation accuracy of 0.956. Somewhat surprisingly, it still outperforms both PaPrBaG and BLAST. It is important to note that for all the other methods, the validation accuracy was much lower than test accuracy. It ranged from 0.574 (BLAST) to 0.784 (RC-LSTM) and 0.787 (RC-CNN+LSTM). We hypothesize that this is because the species randomly assigned to the training set are more related closely to the species in the test set than to those in the validation set. We discuss this problem further in the section 4.1.

The prediction speed is difficult to compare, as PaPrBaG can only run on a CPU, and our neural networks are most efficiently used on GPUs. Both can be trivially parallelized depending on the number of devices available. We measured the time PaPrBaG needed for a complete prediction task (feature extraction and prediction itself) on a single Intel(R) Xeon(R) E7-4890 v2 CPU. The original version of PaPrBaG was able to classify 107 reads/s, and 47% of that time was used for feature extraction. The DNA-only version classified up to 167 reads/s, and most of the speed-up came from the reduction of feature extraction time (which was 25% of the total elapsed time). The classification performance was nearly identical.

The reverse-complement neural networks were tested on a single RTX 2080 Ti GPU. We measured the time of prediction from sequences already converted to binary tensors, as the conversion step may be performed independently on separate CPU devices. The selected RC-LSTM model can predict up to 1817 reads/s and the RC-CNN reaches a speed of 4010 reads/s. Using this rapid architecture, a million-read sample may be processed in just over 4 minutes even on a consumer-level desktop computer.

Integrating signals contained in both mates significantly boosts performance of all the evaluated methods (Table 2). In this case, the RC-CNN achieves the highest accuracy. Compared to the previous state-of-the-art (PaPrBaG with original settings, single reads only), it cuts the error rate almost in half (21.9% vs. 11.3%).

3.2.2 Single-species samples

The evaluation scenario presented in Table 3 models sequencing a pure, single-species isolate, when it can be assumed that all the reads originate from the same genome. After averaging the pathogenic potentials over whole genomes, BLAST, PaPrBaG, and RC-CNN performed equally well, with just one false positive and one false negative. The RC-LSTM and RC-CNN+LSTM ensemble models predict one more false positive, which is reflected in their drastically lower specificity. This is in line with the

Table 1. Classification performance on single reads. $PaPrBaG_{DNA}$ is a variant of PaPrBaG using DNA-based features only. By default, we use the training set to build the BLAST reference database. $BLAST_{All}$ uses all available strains of the training species. $RC\text{-}CNN_{Read\text{-}val}$ was trained using read-level validation. RC-CNN+LSTM is the average of RC-CNN and RC-LSTM predictions.

Classifier	ACC	PPV	TPR	TNR	AUC	AUPR	Pred.
PaPrBaG	0.781	0.756	0.830	0.732	0.859	0.844	1.000
$PaPrBaG_{DNA}$	0.782	0.752	0.841	0.723	0.861	0.847	1.000
BLAST	0.665	0.845	0.818	0.512	n/a	n/a	0.751
$BLAST_{All}$	0.672	0.835	0.838	0.507	n/a	n/a	0.765
$RC\text{-}CNN_{Read\text{-}val}$	0.790	0.733	0.911	0.668	0.880	0.858	1.000
RC-CNN	0.844	**0.825**	0.874	0.815	0.918	0.918	1.000
RC-LSTM	0.838	0.778	**0.946**	0.730	0.916	0.888	1.000
RC-CNN+LSTM	**0.848**	0.794	0.941	0.756	**0.929**	**0.922**	1.000

Table 2. Classification performance on read pairs

Classifier	ACC	PPV	TPR	TNR	AUC	AUPR	Pred
PaPrBaG	0.822	0.789	0.878	0.765	0.898	0.885	1.000
$PaPrBaG_{DNA}$	0.822	0.785	0.887	0.757	0.901	0.890	1.000
BLAST	0.700	0.841	0.866	0.535	n/a	n/a	0.794
$BLAST_{All}$	0.704	0.829	0.883	0.525	n/a	n/a	0.805
RC-CNN	**0.887**	**0.868**	0.913	0.861	**0.948**	**0.947**	1.000
RC-LSTM	0.866	0.804	**0.967**	0.764	0.936	0.909	1.000
RC-CNN+LSTM	0.878	0.824	0.962	0.794	**0.948**	0.940	1.000

For a detailed description of the classifiers see Table 1.

Table 3. Classification performance on single-species samples. Balanced accuracy (BACC) is the mean of sensitivity (TPR) and specificity (TNR). This set contained 39 pathogens and 6 non-pathogens.

Classifier	ACC	BACC	PPV	TPR	TNR	AUC	AUPR
PaPrBaG	**0.956**	**0.904**	**0.974**	0.974	**0.833**	0.932	**0.988**
$PaPrBaG_{DNA}$	**0.956**	**0.904**	**0.974**	0.974	**0.833**	0.932	**0.988**
BLAST	**0.956**	**0.904**	**0.974**	0.974	**0.833**	n/a	n/a
$BLAST_{All}$	**0.956**	**0.904**	**0.974**	0.974	**0.833**	n/a	n/a
RC-CNN	**0.956**	**0.904**	**0.974**	0.974	**0.833**	**0.940**	**0.989**
RC-LSTM	0.933	0.821	0.950	0.974	0.667	0.897	0.978
RC-CNN+LSTM	0.933	0.821	0.950	0.974	0.667	0.923	0.985

For a detailed description of the classifiers see Table 1.

previous results suggesting that those models are more sensitive, even though they do not manage to limit the number of false negatives to zero. Note that the set contained 39 pathogen and only 6 non-pathogen genomes. Validation balanced accuracy ranged from 0.808 (All-strains BLAST) to 0.891 (BLAST, RC-LSTM). Interestingly, only the RC-LSTM and RC-CNN+LSTM ensemble models achieved higher balanced accuracy on the validation set than on the test set. This may reflect the different evolutionary distances between the members of each of the sets (see section 3.2.1).

3.3 Temporal hold-out

For the temporal hold-out test, we predicted labels for reads originating from *Pantoea brenneri*, *Pantoea septica*, and *Pantoea conspicua*, which were labeled pathogenic based on the metadata extracted from the IMG/M database. They were not available when we prepared our training, validation, and primary test sets. Species-wide, the labels were predicted correctly for all three genomes by both BLAST and the reverse-complement networks (Table 4). The RC-LSTM and the RC-CNN+LSTM ensemble model performed best, but the RC-CNN outperformed BLAST as well. PaPrBaG was not used here, as it consistently underperformed in comparison to the deep learning approaches in the previous tests.

Table 4. Temporal hold-out, read pairs

Classifier	ACC	PPV	TPR	Pred.
BLAST	0.881	1.000	0.881	0.883
$BLAST_{All}$	0.895	1.000	0.895	0.896
RC-CNN	0.984	1.000	0.984	1.000
RC-LSTM	**0.992**	1.000	**0.992**	1.000
RC-CNN+LSTM	**0.992**	1.000	**0.992**	1.000

For a description of the classifiers see Table 1.

Table 5. Performance on real data, read pairs

Classifier	ACC	PPV	TPR	Pred.
BLAST	0.746	1.000	0.746	0.750
$BLAST_{All}$	0.787	1.000	0.787	0.790
RC-CNN	0.900	1.000	0.900	1.000
RC-LSTM	**0.984**	1.000	**0.984**	1.000
RC-CNN+LSTM	0.982	1.000	0.982	1.000

For a description of the classifiers see Table 1.

3.4 Real-data results

One of the defining characteristics of real sequencing runs is that the reads generated are not all of the same length. Since the neural networks require a constant input length, we pad the sequences shorter than 250nt with zeroes (interpreted as Ns). If any sequence is longer than this threshold, it is trimmed. This may be a problem for the CNNs with average pooling, as multiple zero-entries significantly lower the activations after a pooling layer. Therefore, we expected those architectures to achieve lower accuracy than the RC-LSTMs on a real dataset. Note that this problem does not apply to RC-CNNs with max pooling, However, those performed worse than their average-pooling counterparts in the tuning and validation step. As they essentially work as motif detectors, we suspect that they are more prone to overfitting. We present the results in Table 5.

Although the RC-CNN suffers from lower accuracy compared to the RC-LSTM and RC-CNN+LSTM ensemble models, it still outperforms BLAST by a large margin. Based on the prediction speed we can name the RC-CNN our *rapid* model, and the RC-LSTM our *sensitive* model. Their predictions may be aggregated with the RC-CNN+LSTM ensemble model, with almost no additional computation, for a boost in sensitivity at a smaller precision cost (see section 3.2.1).

3.5 BacPaCS dataset results

We present the results of evaluation performed on the BacPaCS dataset in Table 6. The RC-networks outperform all of the other methods in terms of balanced accuracy. PaPrBaG's specificity is much higher after retraining (see section 2.4.3). This is also reflected in balanced accuracy, which in this case is even higher than BacPaCS's. The RC-CNN architecture selected based on our primary dataset turns out to be the most specific architecture when trained on the BacPaCS dataset. As expected, the best overall performance is achieved with batch normalization and without input dropout (see sections 2.1.4 and 2.1.7).

Table 6. Performance on the BacPaCS dataset. BACC is the mean of TPR and TNR.

Classifier	TPR	TNR	BACC
BacPaCS	0.92	0.48	0.70
Pathogenfinder	0.63	0.28	0.46
PaPrBaG (Barash *et al.*)	**1.00**	0.05	0.53
PaPrBaG (retrained)	0.72	0.83	0.77
RC-CNN	0.70	**0.93**	0.81
RC-LSTM	0.75	0.85	0.80
RC-CNN+LSTM	0.72	0.90	0.81
RC-CNN+BN	0.78	0.90	**0.84**

4 Discussion

4.1 Validation and test performance

Given the difficulty of the task, the reverse-complement networks offer impressive performance when evaluated on the test set. They are able to predict a high-level, abstract phenotype (namely, being a human pathogen) from a single, isolated read or a read pair, without any additional biological context. They also outperformed the traditional deep learning architectures in the tuning and validation step (see section 3.2.1). However, all of the evaluated tools show a noticeable gap between the validation and test accuracy. We assume that this is a result of the species-level division of the data into the training, validation and test sets – it must be more difficult to predict correct labels for the validation data than for the test data. We hypothesize that the species assigned to the test set must be more similar to the training species than the validation species. We suspect that this could be the case because the evolutionary distance between training an validation organisms is higher that between those in training and test sets. This would be therefore a result of treating all the species as separate, independent entities, which is not biologically correct. Thus, the assumption that data are independent and identically distributed is violated.

We resort to the oversimplification of treating species as independent since balancing the read sets taking the evolutionary distances into account is not trivial. The same approach was used by Deneke *et al.* (2017) for PaPrBaG. In turn, Barash *et al.* (2018) use single strains as primary biological entities, which leads to similar problems. Nevertheless, both approaches do achieve satisfactory results. More exact estimates of the classification error could be obtained using nested cross-validation, but this was not computationally feasible – a single training epoch may take up to 6 hours on a state-of-the-art GPU for some of the most demanding architectures. Therefore, we treat BLAST, a sensitive alignment-based method, as a gold standard reflecting the sequence similarities (and assumed evolutionary distances) between the datasets. In addition, we test the classifiers on independent data. Our approach consistently outperforms BLAST and PaPrBaG, shown previously to constitute the state-of-the-art in sequence homology-based and machine learning based pathogen classification (Deneke *et al.*, 2017). It also fares better that BacPaCS, Pathogenfinder, and PaPrBaG when tested on the original BacPaCS data.

4.2 The definition of a pathogen

It is important to define human pathogens in a reliable and consistent way. However, it is also by no means a trivial task. We do not differentiate between the level of danger individual organisms may pose – dangerous, biosafety level 3 select-agents are put in the same category as opportunistic pathogens causing relatively mild and easily-treated infections. Classifying the risk posed by unrecognized DNA at a higher resolution is an interesting avenue for further research; it is however likely to be challenging due to a comparatively small number of very dangerous bacterial species. We recognize that what constitutes a pathogen is an open problem and different definitions may be interesting for different purposes. For processing big datasets, the labels must be extracted automatically using handcrafted rules compatible with a particular source database. It is therefore immensely important to be aware of the underlying assumptions, as they will be reflected in the final classification decisions of a trained model. We note that comparing models trained using incompatible labels may result in unreliable error estimates, especially if organisms for which the labels differ constitute a substantial fraction of the test set. We recommend following the steps of Barash *et al.* (2018) and sharing the metadata describing the training organisms along the trained models. For this study, they are available at https://gitlab.com/rki_bioinformatics/DeePaC.

4.3 A universal framework for RC-constrained classification

DeePaC is easily extensible and may be used as a generalized framework for building reverse-complement neural architectures beyond the applications described here. In principle, any layer can be plugged in wrapper functions similar to those already implemented, generating a reverse-complement variant of the layer. The framework may also be used for multi-class and multi-label classification, as well as regression tasks if relevant. It is not constrained to a pathogenic potential prediction task; any label or value may be used as a prediction target. It may also be used for sequences longer and shorter than the read length used here. For predicting a categorical phenotype directly from NGS reads, we suggest that the target label should be a relatively abstract, general feature. Phenotypes dependent on single genes or small sets of genes will most probably be very difficult to predict from whole-genome sequencing reads. The variant of the RC configuration (*full*, *siamese* or *none*) may be easily switched by the user, so the framework can be also applied to tasks where the RC-constraint is not important. Also RNA sequences can be analysed, provided that they are converted to binary tensors using the appropriate alphabet.

4.4 A workflow for pathogenic potential prediction

The trained models can be used to predict pathogenic potentials for unknown, unrecognized and novel DNA sequences with a simple script. It is also possible to filter a read set based on a user-defined threshold to select only the reads, for which a network yields confident predictions. This may be used as a crude pre-filter, where the reads associated with the pathogenic label the most may be further investigated in downstream analysis. Furthermore, novel (e.g. synthetic) sequences may be evaluated even if they do not originate from an Illumina sequencing run.

5 Conclusions

The universal framework for reverse-complement neural networks enables transformation of traditional deep learning architectures into their RC-counterparts, guaranteeing consistent predictions for any given DNA sequence, regardless of its orientation. This is especially important in the context of a risk-sensitive task of novel pathogen detection. We show that the RC-networks outperform the previous state-of-the-art on both simulated and real sequencing data, accurately predicting a pathogenic phenotype from single, isolated reads and read pairs. The code, the models trained and example configuration files are available at https://gitlab.com/rki_bioinformatics/DeePaC.

Acknowledgements

We are grateful to Stefan Budach and Annalisa Marsico (Max Planck Institute for Molecular Genetics) for their insights which helped us design the first prototype CNN. We also thank Carlus Deneke (German Federal Institute of Risk Assessment) for sharing his expertise regarding PaPrBaG.

We express our gratitude to Melania Nowicka (Max Planck Institute for Molecular Genetics), as well as Tobias Loka and our colleagues at MF1 (Robert Koch Institute) for many fruitful discussions and comments.

Funding

This work was supported by the Elsa Neumann Scholarship of the State of Berlin (JMB) and the German Academic Scholarship Foundation (JMB).

References

Alipanahi, B. *et al.* (2015). Predicting the sequence specificities of DNA- and RNA-binding proteins by deep learning. *Nature Biotechnology*, **33**(8), 831–838.

Altschul, S. F. *et al.* (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**(3), 403–410.

Andrusch, A. *et al.* (2018). PAIPline: pathogen identification in metagenomic and clinical next generation sequencing samples. *Bioinformatics*, **34**(17), i715–i721.

Barash, E. *et al.* (2018). BacPaCS—Bacterial Pathogenicity Classification via Sparse-SVM. *Bioinformatics*.

Budach, S. and Marsico, A. (2018). pysster: classification of biological sequences by learning sequence and structure motifs with convolutional neural networks. *Bioinformatics*, **34**(17), 3035–3037.

Calistri, A. and Palù, G. (2015). Editorial commentary: Unbiased next-generation sequencing and new pathogen discovery: undeniable advantages and still-existing drawbacks. *Clinical Infectious Diseases: An Official Publication of the Infectious Diseases Society of America*, **60**(6), 889–891.

Carter, S. R. and Friedman, R. M. (2015). DNA Synthesis and Biosecurity: Lessons Learned and Options for the Future.

Chen, I.-M. A. *et al.* (2019). IMG/M v.5.0: an integrated data management and comparative analysis system for microbial genomes and microbiomes. *Nucleic Acids Research*, **47**(D1), D666–D677.

Chollet, F. and others (2015). *Keras*.

Deneke, C. *et al.* (2017). PaPrBaG: A machine learning approach for the detection of novel pathogens from NGS data. *Scientific Reports*, **7**, 39194.

Frank, C. *et al.* (2011). *Epidemic Profile of Shiga-Toxin–Producing Escherichia coli O104:H4 Outbreak in Germany*. Massachusetts Medical Society.

Greenside, P. *et al.* (2018). Discovering epistatic feature interactions from neural network models of regulatory DNA sequences. *Bioinformatics*, **34**(17), i629–i637.

He, K. *et al.* (2015). Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification. *arXiv:1502.01852 [cs]*.

Holtgrewe, M. (2010). Mason – A Read Simulator for Second Generation Sequencing Data. *Technical Report FU Berlin*.

Hong, C. *et al.* (2014). PathoScope 2.0: a complete computational framework for strain identification in environmental or clinical sequencing samples. *Microbiome*, **2**(1), 33.

Ioffe, S. and Szegedy, C. (2015). Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. *arXiv:1502.03167 [cs]*.

Kelley, D. R. *et al.* (2016). Basset: learning the regulatory code of the accessible genome with deep convolutional neural networks. *Genome Research*, **26**(7), 990–999.

Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980 [cs]*.

Langmead, B. and Salzberg, S. L. (2012). Fast gapped-read alignment with Bowtie 2. *Nature methods*, **9**(4), 357–359.

Lecuit, M. and Eloit, M. (2014). The diagnosis of infectious diseases by whole genome next generation sequencing: a new era is opening. *Frontiers in Cellular and Infection Microbiology*, **4**, 25.

Li, H. and Durbin, R. (2009). Fast and accurate short read alignment with Burrows-Wheeler transform. *Bioinformatics (Oxford, England)*, **25**(14), 1754–1760.

Lindner, M. S. and Renard, B. Y. (2015). Metagenomic Profiling of Known and Unknown Microbes with MicrobeGPS. *PLOS ONE*, **10**(2), e0117711.

Lindner, M. S. *et al.* (2017). HiLive: real-time mapping of illumina reads while sequencing. *Bioinformatics*, **33**(6), 917–319.

Locey, K. J. and Lennon, J. T. (2016). Scaling laws predict global microbial diversity. *Proceedings of the National Academy of Sciences of the United States of America*, **113**(21), 5970–5975.

Loka, T. P. *et al.* (2018). Reliable variant calling during runtime of Illumina sequencing. *bioRxiv*, page 387662.

Manara, S. *et al.* (2018). Whole-genome epidemiology, characterisation, and phylogenetic reconstruction of Staphylococcus aureus strains in a paediatric hospital. *Genome Medicine*, **10**(1), 82.

Martín Abadi *et al.* (2015). *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*.

National Academies of Sciences, Engineering, and Medicine (2018). *Biodefense in the Age of Synthetic Biology*. The National Academies Press.

National Research Council (2010). *Sequence-Based Classification of Select Agents: A Brighter Line*. The National Academies Press.

Nielsen, A. A. K. and Voigt, C. A. (2018). Deep learning to predict the lab-of-origin of engineered DNA. *Nature Communications*, **9**(1), 3135.

Onimaru, K. *et al.* (2018). A regulatory-sequence classifier with a neural network for genomic information processing. *bioRxiv*, page 355974.

Piro, V. C. *et al.* (2018). ganon: continuously up-to-date with database growth for precise short read classification in metagenomics. *bioRxiv*, page 406017.

Poplin, R. *et al.* (2018). A universal SNP and small-indel variant caller using deep neural networks. *Nature Biotechnology*, **36**(10), 983–987.

Quang, D. and Xie, X. (2016). DanQ: a hybrid convolutional and recurrent deep neural network for quantifying the function of DNA sequences. *Nucleic Acids Research*, **44**(11), e107–e107.

Rosen, G. *et al.* (2008). Metagenome Fragment Classification Using N-Mer Frequency Profiles.

Rosen, G. L. *et al.* (2011). NBC: the Naïve Bayes Classification tool webserver for taxonomic classification of metagenomic reads. *Bioinformatics*, **27**(1), 127–129.

Shrikumar, A. *et al.* (2017). Reverse-complement parameter sharing improves deep learning models for genomics. *bioRxiv*, page 103663.

Srivastava, N. *et al.* (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, **15**, 1929–1958.

Truong, D. T. *et al.* (2015). MetaPhlAn2 for enhanced metagenomic taxonomic profiling. *Nature Methods*, **12**(10), 902–903.

Wattam, A. R. *et al.* (2017). Improvements to PATRIC, the all-bacterial Bioinformatics Database and Analysis Resource Center. *Nucleic Acids Research*, **45**(D1), D535–D542.

Willis, A. (2016). Extrapolating abundance curves has no predictive power for estimating microbial biodiversity. *Proceedings of the National Academy of Sciences*, **113**(35), E5096–E5096.

Wood, D. E. and Salzberg, S. L. (2014). Kraken: ultrafast metagenomic sequence classification using exact alignments. *Genome Biology*, **15**(3), R46.

Zeng, H. *et al.* (2016). Convolutional neural network architectures for predicting DNA–protein binding. *Bioinformatics*, **32**(12), i121–i127.

Zhou, J. and Troyanskaya, O. G. (2015). Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, **12**(10), 931–934.