1 Using Deep Learning for Gene Detection and Classification in

2 Raw Nanopore Signals

3

4 Marketa Nykrynova^{1,*}, Vojtech Barton^{1,b}, Roman Jakubicek^{1,c}, Matej Bezdicek^{2,d},
5 Martina Lengerova^{2,e}, Helena Skutkova^{1,f}

6 ¹ *Department of Biomedical Engineering, Faculty of Electrical Engineering, Brno*
7 *University of Technology, Brno 61600, Czechia*

8 ² *Department of Internal Medicine – Hematology and Oncology, University Hospital*
9 *Brno, Brno 62500, Czechia*

10

11 ^aORCID: 0000-0002-0205-6935.

12 ^bORCID: 0000-0002-1918-2989.

13 ^cORCID: 0000-0003-4293-260X.

14 ^dORCID: 0000-0002-5833-8325.

15 ^eORCID: 0000-0001-8739-9998.

16 ^fORCID: 0000-0003-4562-2746.

17

18

19

20

21

22

23

24

25

26

27 **PREPRINT - Manuscript was submitted to the journal Genomics, Proteomics**
28 **and Bioinformatics on 15th December 2021.**

29

30 **Abstract**

31 Recently, nanopore sequencing has come to the fore as library preparation is rapid and
 32 simple, sequencing can be done almost anywhere, and longer reads are obtained than
 33 with next-generation sequencing. The main bottleneck still lies in data postprocessing
 34 which consists of basecalling, genome assembly, and localizing significant sequences,
 35 which is time consuming and computationally demanding, thus prolonging delivery of
 36 crucial results for clinical practice. Here, we present a neural network-based method
 37 capable of detecting and classifying specific genomic regions already in raw nanopore
 38 signals – squiggles. Therefore, the basecalling process can be omitted entirely as the
 39 raw signals of significant genes, or intergenic regions can be directly analysed, or if
 40 the nucleotide sequences are required, the identified squiggles can be basecalled,
 41 preferably to others. The proposed neural network could be included directly in the
 42 sequencing run, allowing real-time squiggle processing.

43

44 **KEYWORDS:** Nanopore sequencing; Squiggles; Neural network; MLST; Bacterial
 45 typing

46

47

48 **Introduction**

49 DNA sequencing technologies revolutionized our ability to study genetic variations at
50 the molecular level, which is necessary for a broad spectrum of applications from
51 bacterial typing to cancer research. The introduction of the latest technology -
52 nanopore sequencing - was a major breakthrough [1]. Nanopore sequencing, unlike
53 other methods, does not require DNA synthesis or amplification. Sample preparation
54 is significantly more straightforward, and sequencing can be performed practically
55 anywhere [2]–[4]. The technology uses small pores located in a membrane to which a
56 voltage is applied. When the DNA strand passes through the pore, the electric
57 current's changes are measured, and as a result, the signal of the read called a
58 'squiggle' is obtained.

59 Besides the advantages of easy preparation and simple use, nanopore sequencing
60 also provides reads with lengths from tens to hundreds of kilobase pairs (kbp), with a
61 record of more than 2 Mbp [5]. However, the main bottleneck in this technology lies
62 in the low read accuracy, which is improving with each chemistry update, yet still
63 does not compare with second-generation sequencers. Inaccuracies also emerged
64 during the basecalling process, where the squiggles are converted to nucleotide
65 sequences. The basecallers achieve an average read accuracy of 85-95 % [6], which is
66 insufficient for analyses such as single nucleotide variant detection. Although using
67 high genome coverage in post-sequencing analysis, the consensus accuracy can be
68 improved to 99.9 % [7]. If the basecalling process was bypassed and an analysis of the
69 raw squiggles was performed, the results would be more precise and delivered in a
70 shorter time as the Oxford Nanopore Technologies MinION sequencing platform
71 allows real-time access to the sequencing run [8]. Thus, waiting for the whole run to
72 finish would not be necessary, and the crucial information could be analysed almost in
73 real-time. As the squiggle analysis can be performed during the sequencing itself, and
74 the sequencing process can be stopped based on the real-time outputs from the
75 analytical software, there is no need to use a flowcell's whole sequencing capacity.

76 As mentioned above, the basecalling process has lower read accuracy caused by
 77 several problems. Firstly, the change in current deviation does not correspond to one
 78 nucleotide passing through the nanopore, but approximately five nucleotides, leading
 79 to $4^5 = 1024$ current levels [9]. Moreover, the number of possible current levels can be
 80 higher because the 5-nucleotide step and the speed of the nucleotide strand passing
 81 through the nanopore are not constant. Secondly, the same bases can be chemically or
 82 epigenetically modified (e.g. 5-methylcytosine) [10], and the whole signal is also
 83 affected by noise. These complications mean that only neural networks (NNs) are
 84 used for basecalling nowadays [10]. NNs have also found applications in other parts
 85 of nanopore sequencing, such as selective sequencing. An example of a tool used to
 86 determine whether to eject the sequenced molecule or continue sequencing could be
 87 SquiggleNet [11], which uses a convolutional neural network learned from the
 88 reference organism's sequencing data. The classifier then decides the sequenced
 89 segment's location and whether to continue sequencing or eject the molecule.
 90 Recently, NNs were also employed to distinguish mitochondrial DNA from genomic
 91 DNA in squiggles during a sequencing run [12].

92 With NNs, it would be possible to identify specific genomic regions in raw
 93 nanopore data without the need for basecalling, as shown in the presented article.
 94 Even more genes or gene fragments can be found once the neural network can
 95 recognize and classify them. The proposed NN was used to predict and classify seven
 96 multilocus sequence typing (MLST) loci (*gapA*, *infB*, *mdh*, *pgi*, *phoE*, *rpoB*, *tonB*) in
 97 29 *Klebsiella pneumoniae* genome squiggles. Raw squiggle analysis can bring more
 98 precise information, as the epidemiological and chemical modification can also be
 99 studied. In the future, distinguishing bacterial strains could be done using only the
 100 signals with no basecalling, providing crucial epidemiological information for early
 101 outbreak identification.

102

103 **Results**

104 **Analysed genes**

105 The seven MLST loci (*gapA*, *infB*, *mdh*, *pgi*, *phoE*, *rpoB*, *tonB*) from BIGSdb [13]
 106 were chosen for showing the proposed neural network's performance. The first allele
 107 from each MLST gene was searched for in basecalled reads. The median lengths of
 108 the located MLST genes were, on average, 4 bp shorter than the queries. Thus, most
 109 of the identified genes contained almost the whole allele sequence. The analysed
 110 sequence variability was about 6.31% (**Table 1**). This variability is caused mainly by
 111 the presence of other alleles from a given gene in the dataset, as only one allele for
 112 each gene was searched for. Based on these values, it can be said that the dataset is
 113 variable enough to train the network to be able to recognize different gene's alleles.

114 **Dataset preparation**

115 Gene signals that could be used to train the neural network and later validate its
 116 performance were obtained by the following described process. BLAST (2.9.0+, [14])
 117 was employed to localize sequences of interest. The fast5 files containing basecalled
 118 fastq sequences were used to prepare the signal database. Then, the gene templates
 119 were examined in basecalled data, and results (hits) were saved in CSV format. In
 120 addition, the whole squiggles containing the gene sequences were extracted and saved
 121 in an internal h5 format, so they could be swiftly accessible and easily modified.

122 If more genomes were sequenced in one run, the BLAST results contained hits for
 123 all genomes from a particular run. To assign barcodes to the hits, demultiplexing
 124 tables were created for sequencing runs where barcoding kits were used. The outputs
 125 from the Guppy barcoder were used for this purpose. Each created table contained all
 126 reads ID from the sequencing run and their corresponding barcodes; so, it was
 127 possible to add sample identification to the hits in the BLAST table.

128 The BLAST results were filtered to remove random and partial hits. For further
 129 processing, only the hits with a percentage of identical matches more significant or
 130 equal to 90 %, the length at least 90 % of query length, and the e-value lower or equal
 131 to 1e-50 were chosen. The searched sequences were found on both the leading and
 132 complementary strands.

133 In the last signal extraction step, the gene sequences' BLAST coordinates were
 134 recalculated to signal coordinates. Thus, a dataset containing squiggles with desired

135 genes was created. To each squiggle, the gene signal coordinates were added. The
136 neural network should also recognize squiggles without genes; therefore, datasets with
137 no genes were created.

138 For preprocessing the raw sequencing fast5 files and dataset preparation, the
139 internally developed MANASIG [15] package was used and is available on [GitHub](#).

140 **Signals for training/validating neural network**

141 In total, 48,860 squiggles from which 38,867 contained one of the seven
142 housekeeping genes fragments were analysed. The length of squiggles with genes
143 ranged from 4,362 to 4,477,607 samples with a median of about 142,869 samples.
144 The median gene fragment lengths were 4,815 (*gapA*), 3,417 (*infB*), 5,197 (*mdh*),
145 4,643 (*pgi*), 4,586 (*phoE*), 5,387 (*rpoB*) and 4,565 (*tonB*) samples. The number of
146 signals with no genes was 9,993 and their lengths varied from 2,008 to 1,411,573
147 samples with a median value of 24,408 samples. These squiggles were randomly
148 generated from the seven sequencing runs. For a detailed number of squiggles from
149 each analysed genome, see Table S1.

150 For neural network training, about 75 % of all squiggles were used, and the rest
151 were used to validate NN performance.

152 **Gene detection in squiggles**

153 From the validation dataset, 41 squiggles with corrupted gene coordinates were
154 removed. In total, 11,887 squiggles were used for validation, of which 2,000 had no
155 gene sequences. The representation of individual genes in the dataset was as follows:
156 *gapA* – 1,671, *infB* – 1,570, *mdh* – 1,644, *pgi* – 1,659, *phoE* – 1,160, *rpoB* – 1,179 and
157 *tonB* – 1,004.

158 The gene predictions in the downsampled squiggles containing any genes of
159 interest were classified into two categories – a gene found, and a gene not found. To
160 evaluate, the calculated dice coefficients were used. On average, the detection success
161 rate was about 98%; see **Table 2** for specific values. The gene found category was
162 further split into the other two subgroups – the gene was found correctly (dice $\geq$ 0.9),
163 or the gene was shifted. If the predicted gene was labelled as shifted, four variants
164 could be observed – the detected part was inside/outside the annotated borders or

shifted before/beyond the start/end of the gene. Examples of all possible gene detection cases are shown in **Figure 1**, and squiggle percentages in the given categories can be seen in **Table 3**. The prediction with dice greater than or equal to 0.9 differed from the annotation position by, on average, about ten samples. Predictions with lower dice consisted mainly of genes shifted beyond the annotated gene start and end, and from predictions inside annotated gene boundaries.

In the case of squiggles with no genes, predictions with dice equal to one were marked as correct. In total, 96.6% of squiggles without any genes were successfully recognized, and no gene was detected in them.

Gene squiggles classification

The squiggles from the validation dataset were classified via the proposed neural network into eight categories, where seven categories were for analysed genes, and the last one was for the squiggles with no gene. For each squiggle, the probabilities that the gene belongs to a given category were established. From these values, the maximum was chosen, and the squiggles were assigned to a given category.

The squiggles with no MLST genes were correctly classified in 99.80% of cases and only four squiggles were misclassified. The percentage of successful gene classification was about 94.67% for five out of seven MLST loci. In the case of *pgi*, the success rate dropped to 87.64% and in the case of *mdh* to 83.33%. See the detailed results in **Figure 2**.

The confusion matrix shows the true positive rate, which is the highest for the squiggles with no detectable gene and in the case of MLST genes, the highest value of 95.76% is for *rpoB* gene. In general, the normalized true positive value for all MLST genes was 92.04%; if squiggles without genes are included, the average true positive rate is even higher, at 93.01%.

On the other hand, the false discovery rate was the highest for the squiggles without genes. The difference between the average false discovery rate for the squiggles with and without genes is 13.65%. It can be concluded that there is a much higher rate of not identifying the gene than identifying it incorrectly.

195 Discussion

196 A comparison of detected and annotated gene coordinates (**Figure 3**) and their
197 overlaps was conducted. The results showed that they came from the same statistical
198 distribution. The neural network tended to identify the genes a bit longer than
199 annotated. This phenomenon can be caused by the signal length irregularity in the
200 same sequence. Also, the signals were downsampled for the neural network. This
201 pre-processing and the need to recompute the positions of genes to the original space
202 signal can cause some variations in the genes' signals lengths.

203 The sequencing data are stored in the fast5 format, which is based on hdf5 format.
204 However, hdf5 is not fully backwards compatible; the hierarchical structure can
205 change with every new nanopore chemistry or software upgrade, and no complex
206 descriptions about the fast5 nanopore format and its modification exist. For this
207 reason, our proposed package for fast5 file processing can be used with R9.4.1
208 flowcells, two sequencing kits (SQK-RBK004, SQK-LSK109) and nanopore
209 MinKNOW v19 and v20 sequencing software; with other versions, it might be
210 necessary to modify the package. Completely new chemistry can also influence signal
211 properties such as squiggle lengths, and in that case, new neural network training
212 would be needed. Also, the neural network may not work correctly if the squiggle
213 lengths in the training/validating dataset contain outliers such as extremely long ones.

214 During the gene coordinates' detection in predicted signals, it was found that gene
215 occurrence functions sometimes contained more than just one peak exceeding a
216 specified threshold, as shown in **Figure 4A** and **Figure 4B**. Multiple predictions were
217 observed in 3.81% squiggles from the validation dataset, and in the majority of cases
218 (more than 90%), two peaks were predicted. After detecting peak coordinates, the
219 corresponding basecalled nucleotide sequences were analysed. It was found out that
220 the false positive predictions contained sequences with a partial match to the desired
221 gene, see **Figure 4C**. Nevertheless, the partial matches were significantly shorter than
222 the examined sequences; therefore, they could be filtered in postprocessing. However,
223 there could be a problem with setting the filter parameters because the partial matches

may have a different numbers of samples in the squiggles as the speed of DNA passing through the pore is not constant and the sampling is non-equidistant. This mentioned multiple detection problem could be solved if the genes or other specific sequences we wanted to search for are unique and non-repetitive in the analysed bacterial species. On the other hand, from multiple detection results, it can also be said that the neural network could find even short signals that corresponded to several dozen base pair long sequences, such as parts of genes. In addition, it could recognize the squiggles even if there were many mutations; thus, the network can be used to predict and classify even highly variable genomic regions.

If more than one gene is located in the analysed squiggle, it could cause a problem for proper neural network function. Multiple gene presence may occur if long squiggles are produced during the sequencing process. To avoid this problem, genomic regions we wanted to detect should be carefully selected. The first option is to choose genes that have sufficient distance from each other in the analysed species. The second option is to pre-process the squiggles before sending them into a neural network. Signals longer than a given threshold can be divided into several shorter signals, ensuring that only one targeted region will be present in each signal.

Conclusion

This paper presented a deep learning method to identify and classify specific genomic regions in raw nanopore sequencing data. The proposed neural network can be used to find whole genes, their parts or intergenic regions. We showed one of the possible neural network uses – detecting and classifying seven MLST loci in *K. pneumoniae* genomes in squiggles.

The percentage of correctly predicted genes was 98.2%, and they were successfully classified in 92.9% cases. The squiggles with no MLST loci were correctly predicted in 96.6% and classified in 99.8% cases. The NN achieved the same accuracy as basecalling tools, and if postprocessing was employed, the accuracy could be even higher.

253 The main requirement of the proposed approach for gene prediction and
254 classification is a large amount of data to train the neural network. Without sufficient
255 data, the network would not be adequately trained, and precise results could not be
256 obtained. From the study, it can also be said that the genomic regions detected via the
257 neural network should be unique, non-repetitive sequences of any length and should
258 be located in the analysed genomes with sufficient distances between them.

259 Nanopore sequencing has huge potential in routine clinical practice. It can be used
260 instead of time consuming NGS to deliver crucial epidemiology information earlier.
261 For example, if there is a need to analyse samples from one hospital department to
262 find out if there is an outbreak or not, if an infection is spreading via instruments,
263 patients or medical staff, nanopore sequencing can be employed. NGS analysis would
264 consist of library preparation, sequencing, and in silico sequence type determination
265 from post-processed sequencing data, which can take two to three days while the
266 potential epidemic spreads uncontrollably. The results could be delivered the same
267 morning if nanopore sequencing combined with the proposed NN is employed. The
268 squiggles containing MLST loci can be recognized during a run or immediately after
269 it and analysed. If more genomes are sequenced in one run, the barcodes attached to
270 MLST squiggles can be used to find out which, e.g. *gapA* loci belongs to which
271 genome.

272 The network can be trained to recognize different significant sequences. Hence,
273 the proposed approach can be used for other purposes, such as direct positive clinical
274 sample sequencing. The network can filter out the human sequences, identify MLST
275 loci, and determine infectious agents' sequence types. The cultivation will not be
276 needed, which significantly saves time.

277

278 **Materials and methods**

279 **Analyzed bacterium**

280 *K. pneumoniae* is a Gram-negative opportunistic pathogen from the
281 *Enterobacteriaceae* family. Usually, it affects immunocompromised patients, and the

majority of *K. pneumoniae* infections are hospital-acquired. The gastrointestinal tract colonization generally occurs before nosocomial infections develop, which usually affect the urinary tract, respiratory tract or result in septicemia or soft tissue infection [16],[17]. The genome size of *K. pneumoniae* is about 5.5 Mbp and incorporates about 5,000 to 6,000 genes from which 2,000 genes form the core genome, and almost 30,000 genes are parts of the pangenome [18].

Genome sequencing

In this article, 29 *K. pneumoniae* isolates collected between 09/2014 and 07/2019, mainly at the Department of Internal Medicine, Hematology and Oncology at the University Hospital Brno, were analysed. The high molecular weight DNA was extracted using the MagAttract HMW DNAKit (Qiagen, Venlo, NL), and the NanoDrop (Thermo Fisher Scientific, Waltham, MA, USA) was employed to measure the purity of the extracted DNA. The DNA concentration was checked by Qubit 3.0 Fluorometer (Thermo Fisher Scientific, Wilmington, DE, United States) and using Agilent 4200 TapeStation (Agilent Technologies, Santa Clara, CA, USA) the proper length of the isolated DNA was checked. The Rapid Barcoding Kit (Oxford Nanopore Technologies, Oxford, UK) was used to prepare the sequencing library for 27 *K. pneumoniae* isolates. For the remaining two samples, the Ligation Sequencing 1D Kit (Oxford Nanopore Technologies, Oxford, UK) was used to prepare the library for Oxford Nanopore sequencing. The sequencing was performed using the MinION sequencing platform (Oxford Nanopore Technologies, Oxford, UK) with R9.4.1 flowcells.

The sequenced genomes were basecalled and, in the case of the pooled library, separated according to barcodes using Guppy software (3.4.4+a296acb). The data quality was checked using PycoQC (v2.2.3, [19]). See Table S2 for detailed information about each sequencing run. The analyzed datasets can be found in the National Center for Biotechnology Information Sequence Read Archive database under a BioProject with accession number [PRJNA786743](https://www.ncbi.nlm.nih.gov/bioproject/PRJNA786743).

Neural network architecture

311 To sort out the task of enabling gene localization and classification, some challenges
312 had to be overcome. The main problem was squiggles that were too long, moreover
313 with uneven sampling time. This distinctly increases computational complexity and
314 memory demands, and first and foremost, causes a significant vanishing gradient
315 (VG) effect during training, especially for the recurrent nets. Further, conventional
316 neural networks cannot be applied to signals of different lengths, and they require
317 strictly the same input length.

318 The proposed NanoGeneNet can be divided into three basic parts: feature
319 extraction, gene localization (so-called a sequence-to-sequence regime) and finally,
320 gene classification with another feature extraction (a sequence-to-vector regime).
321 Using the combination of convolution and recurrent networks turned out to be a great
322 solution for long and uneven signal lengths. For more details on architecture design,
323 see **Figure 5**.

324 The network was realized in Python 3.9 with the PyTorch library. The source code
325 for training, validation, and especially feedforward for NanoGeneNet is available on
326 [GitHub](#) along with a demo and an example squiggle.

327 *Local feature extraction with downsampling*

328 The encoder part of the net allows local features to be extracted from the raw signal
329 using a recursive repeating block. As shown in **Figure 5**, the block contains two
330 convolution layers (Conv) with a kernel size of 3 and depth of 16, doubled in each
331 recursion. The Conv layer is always followed by a ReLU activation function, and the
332 block ends with a batch normalization layer and nonlinear spatial reduction layer -
333 MaxPooling. This encoder part occurs twice in the NanoGeneNet.

334 The network part designed in this way provides a new local feature signal
335 reflecting the occurring or non-occurring gene from an input signal. Due to the
336 MaxPooling layer, there is a spatial reduction in the signal length of M with a
337 sub-sampling factor of 2^4 to decrease computational complexity, while the only
338 relevant features for gene detection are retained during net feedforward. Since this
339 part of the network only views a very local part of the signal, the following LSTM
340 (Long Short-Term Memory) [20] network has the task of viewing the whole signal

globally and determining the local features for each signal sample, considering all previous samples (long memory). The output is then the tensor of local feature signals sized $M \times 256$, where M is the length of the downsampled feature signals.

Due to a recurrent LSTM net, the input can be different length signals and sub-sampling mildly suppresses the VG effect and significantly decreases GPU memory demands. The complete suppression of VG problem is performed by multistage training.

Gene occurrence prediction

From the previous LSTM block, each downsampled output signal sample is now coded by feature vector, which inputs into a two-class classification part. Here, two fully connected (FC) networks with ReLU and Softmax activation function can be used, respectively. This part of the proposed NanoGeneNet classifies each downsampled signal sample into two classes; gene or non-gene. The output is tensor $M \times 2$ defining probability assignment to all classes. Since there are two classes, we can use the probabilities of only the first class as an output gene occurrence function. Examples of such functions are shown in **Figure 1**.

Gene classification

In this part, the local feature tensor from the previous encoder and LSTM is concatenated with the obtained likelihood function and the resulting tensor is $M \times 257$ in size. The tensor is encoded into a new feature tensor corresponding with the feature import to classify gene types via a second encoder block with a sub-sampling factor of 2^3 . Then using another LSTM network, the new local feature tensor is transformed into another new tensor (1×512) reflecting only the whole signal's global features to enable classification of the whole signal into gene class. This task is performed via the last part of NanoGeneNet - the multiclass classification network. It has the same architecture as the previous 2-class net, except that the last FC layer has eight neurons followed by SoftMax, enabling a one-hot coding of output class probabilities. The final decision is based on the choice of class with the maximal value.

Network training details

Training and validation dataset

371 The available signal dataset was divided on the genome level by hand to ensure data
372 distribution consistency for training and validation. For the validation dataset of each
373 gene, all randomly selected genome signals were selected to make up something
374 between 20 % and 30 % of the entire database. It cannot be done exactly because each
375 file contains a different number of signals for specific genes within each genome. As
376 the genome includes the same gene sequence, all signals always within the genome
377 were used for training/testing; ignoring the uneven time sampling, i.e., a nonlinear
378 scaling, which is taken as a kind of augmentation useful for training.

379 *Multistage approach*

380 Generally, the VG problem is related to the signal length, the longer the signal, the
381 greater the effect of this problem. Therefore, multistage training was proposed. In the
382 first phase, the first encoder with sub-sampling was pre-trained on shorter signals. In
383 each iteration, the current signal was randomly cut with random termination and
384 random length in a range of 20-80 thousand samples. In the case of gene signal
385 occurrence, this cut training signal always contained the whole gene. In this way, the
386 encoder and LSTM1 training were achieved with a minimal VG problem. To derive a
387 loss function during training, the 2-class classification was trained concurrently based
388 on cut annotations in this way.

389 In the second phase, the first encoder was additionally retrained on the database
390 containing the non-gene signals. Finally, in this way, the pre-trained net (encoder,
391 LSTM1 and 2-class net) was fine-tuned for signals with genuine length. The same
392 training strategy was also used for the second part of NanoGeneNet, but in this case
393 for an eight-class whole-signal classification task and with its first part frozen.

394 *Training parameters*

395 All training dataset signals were randomly shuffled within each new iteration, and all
396 nets were trained from scratch. As both tasks were defined as a classification with a
397 class probability estimation, a weighted cross-entropy was chosen as a loss function.
398 Further, an Adam algorithm (Adaptive Moment Estimation) [21] was used as an
399 optimization algorithm, and the initialization Learning Rate (LR) was set to 0.001,
400 and the weight decay to 0.0001. During all stages of training, the LR was manually

changed (decreased) based on the designer's experiences. A batch size greater than one could only be used in the first training stage, where the net worked with cut signals; here, the batch size was 8, and a larger batch size resulted in too much regularization. In the training phase, the Dropout layer [22] worked with a probability of 50%. Other hyperparameters were set by default in PyTorch, which can be found in the shared source code on [GitHub](#).

Hardware requirements

NanoGeneNet training was performed on a computational device with Intel Xeon E5-2603v4, 16 GB RAM and nVidia Titan Xp, 12 GB GDDR5 graphics card. The network was realized in Python 3.9 with the PyTorch library. One stage of training took around two hours, and required up to 10 epochs for a sufficient success rate.

The minimal hardware requirements depended on the input squiggle length, where the above-mentioned device had sufficient parameters to process our database. The prediction and classification of a squiggle with median of length (143 thousand samples) took around 0.2 sec in total. There is a linear dependence between the signal length and computational time, where the longest squiggle from our database (almost 4.5 million samples) took around four seconds.

Gene coordinates detection in predicted signals

The outputs from the first part were gene occurrence functions determining the probability for each sample in the downsampling signal being part of the gene. The positions of the predicted gene were detected to evaluate the prediction accuracy. In each gene occurrence curve, the peaks exceeding a threshold of 0.9 were found and further analysed as possible target genes. The shortest possible distance between two peaks was set to 100 samples; otherwise, the peaks were merged into one. Boundaries where the gene occurrence curve exceeded 0.5 were located around possible gene peaks in the next step. Then the gene boundaries were extended sample by sample to find a precise prediction beginning and end. The samples were added to the gene until the difference between the two adjacent samples was higher than 0.1.

For each calculated gene's coordinates, the dice coefficient was calculated as $DSC = 2TP / (2TP + FP + FN)$,

431 where TP was the number of samples correctly classified as gene, the FP was the
 432 number of samples falsely labelled as gene, and FN was the number of samples
 433 falsely labelled as not-gene. If just one peak was observed in the signal, the dice was
 434 calculated for it. In cases where more peaks were detected in the gene occurrence
 435 function, the coefficient was calculated for each of them. For gene prediction and
 436 detection evaluation, the detected gene positions with the highest dice were selected.
 437

438 **Authors' contributions**

439 MN, VB, and HS contributed to the conception and design of the study. MN and VB
440 created and implemented the algorithm for fast5 processing and evaluated the results.
441 RJ designed and implemented the neural network. ML and MB ensured the biological
442 aspects of the project. MN and RJ wrote the manuscript. All authors read and
443 approved the final manuscript.

444

445 **Competing interests**

446 The authors have declared no competing interests.

447

448 **Acknowledgments**

449 This work has been supported by grant FEKT-K-21-6912 realised within the project
450 Quality Internal Grants of BUT (KInG BUT), Reg. No. CZ.02.2.69 / 0.0 / 0.0 /
451 19_073 /0016948, which is financed from the OP RDE. Collecting, processing,
452 storing and sequencing of all bacterial isolates used in this study was supported by the
453 Ministry of Health of the Czech Republic, Grant No. NV19-09-00430, all rights
454 reserved.

455

References

- [1] Kono N, Arakawa K. Nanopore sequencing: Review of potential applications in functional genomics. *Dev Growth Differ.* 2019; 61(5):316–26.
- [2] Hoenen T, Groseth A, Rosenke K, Fischer RJ, Hoenen A, Judson SD, et al. Nanopore Sequencing as a Rapidly Deployable Ebola Outbreak Tool. *Emerg Infect Dis.* 2016; 22(2):331–4.
- [3] Johnson SS, Zaikova E, Goerlitz DS, Bai Y, Tighe SW. Real-Time DNA Sequencing in the Antarctic Dry Valleys Using the Oxford Nanopore Sequencer. *J Biomol Tech.* 2017; 28(1):2–7.
- [4] Castro-Wallace SL, Chiu CY, John KK, Stahl SE, Rubins KH, McIntyre ABR, et al. Nanopore DNA Sequencing and Genome Assembly on the International Space Station. *Sci Rep.* 2017; 7(1):18022.
- [5] Amarasinghe SL, Su S, Dong X, Zappia L, Ritchie ME, Gouil Q. Opportunities and challenges in long-read sequencing data analysis. *Genome Biol.* 2020; 21(1):30.
- [6] Wang Y, Zhao Y, Bollas A, Wang Y, Au KF. Nanopore sequencing technology, bioinformatics and applications. *Nat Biotechnol.* 2021; 39(11):1348–65.
- [7] Rang FJ, Kloosterman WP, de Ridder J. From squiggle to basepair: computational approaches for improving nanopore sequencing read accuracy. *Genome Biol.* 2018; 19(1):90.
- [8] Loose M, Malla S, Stout M. Real-time selective sequencing using nanopore technology. *Nat Methods.* 2016; 13(9):751–4.
- [9] Lu H, Giordano F, Ning Z. Oxford Nanopore MinION Sequencing and Genome Assembly. *Genomics Proteomics Bioinformatics.* 2016; 14(5):265–79.
- [10] Wick RR, Judd LM, Holt KE. Performance of neural network basecalling tools for Oxford Nanopore sequencing. *Genome Biol.* 2019; 20(1):129.
- [11] Bao Y, Wadden J, Erb-Downward JR, Ranjan P, Zhou W, McDonald TL, et al. SquiggleNet: real-time, direct classification of nanopore signals. *Genome Biol.*

2021; 22(1):298.

[12] Danilevsky A, Polsky AL, Shomron N. Real-time selective sequencing using
nanopores and deep learning. 2021; :1–18.

[13] Jolley KA, Maiden MCJ. BIGSdb: Scalable analysis of bacterial genome
variation at the population level. BMC Bioinformatics. 2010; 11(1):595.

[14] Camacho C, Coulouris G, Avagyan V, Ma N, Papadopoulos J, Bealer K, et al.
BLAST+: architecture and applications. BMC Bioinformatics. 2009; 10(1):421.

[15] Barton V, Nykrynova M, Skutkova H. MANASIG: Python Package to
MANipulate NANopore SIGnals from sequencing files. In: 2021 IEEE
International Conference on Bioinformatics and Biomedicine (BIBM). 2021. p.
1–7 (in press).

[16] Choby JE, Howard Anderson J, Weiss DS. Hypervirulent *Klebsiella*
pneumoniae – clinical and molecular perspectives. J Intern Med. 2020;
287(3):283–300.

[17] Martin RM, Bachman MA. Colonization, Infection, and the Accessory Genome
of *Klebsiella pneumoniae*. Front Cell Infect Microbiol. 2018; 8(January):1–15.

[18] Wyres KL, Holt KE. *Klebsiella pneumoniae* Population Genomics and
Antimicrobial-Resistant Clones. Trends Microbiol. 2016; 24(12):944–56.

[19] Leger A, Leonardi T. pycoQC, interactive quality control for Oxford Nanopore
Sequencing. J Open Source Softw. 2019; 4(34):1236.

[20] Hochreiter S, Schmidhuber J. Long Short-Term Memory. Neural Comput.
1997; 9(8):1735–80.

[21] Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. 3rd Int Conf
Learn Represent ICLR 2015 - Conf Track Proc. 2014; :1–15.

[22] Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R. Dropout:
A Simple Way to Prevent Neural Networks from Overfitting. J Mach Learn
Res. 2014; 15(56):1929–58.

515 **Table 1 Statistics about seven analysed MLST loci**

	Query allele length [bp]	Number of alleles in dataset	Identified gene median length [bp]	Variability of identified gene [%]
<i>gapA</i>	450	4	446	6.35
<i>infB</i>	318	6	314	5.57
<i>mdh</i>	477	5	471	6.14
<i>pgi</i>	432	7	428	6.32
<i>phoE</i>	420	11	415	6.59
<i>rpoB</i>	501	7	499	5.81
<i>tonB</i>	414	14	409	7.40

516

517 **Table 2 Percentages of identified or non-identified MLST loci**

[%]	<i>gapA</i>	<i>infB</i>	<i>mdh</i>	<i>pgi</i>	<i>phoE</i>	<i>rpoB</i>	<i>tonB</i>
Gene found	99.04	98.41	97.51	96.08	98.88	98.98	98.61
Gene not found	0.96	1.59	2.49	3.92	1.12	1.02	1.39

518

519 **Table 3 Percentages of identified MLST loci including shifted genes**

[%]	<i>gapA</i>	<i>infB</i>	<i>mdh</i>	<i>pgi</i>	<i>phoE</i>	<i>rpoB</i>	<i>tonB</i>
Gene was found correctly	95.75	93.50	94.71	90.84	93.71	95.17	94.62
Gene was shifted:	3.29	4.90	2.80	5.24	5.18	3.82	3.99
- outside annotated borders	0.06	0.25	0.18	0.48	0.09	0.17	0.60
- inside annotated borders	1.38	1.34	1.58	2.29	1.81	1.27	1.20
- shifted beyond gene start/end	1.68	3.31	0.97	2.35	3.10	2.29	2.19
- shifted before gene start/end	0.18	0.00	0.06	0.12	0.17	0.08	0.00

520

521

522

523

Figures

Figure 1 Examples of gene detection, annotated positions are blue, gene occurrence functions are black and detected gene positions are red
A. gene was found correctly, B. gene was not found, C. gene was outside annotated borders, D. gene was inside annotated borders, E. gene was shifted to right, F. gene was shifted to left.

		predicted class							
		<i>gapA</i>	<i>infB</i>	<i>mdh</i>	<i>pgi</i>	<i>phoE</i>	<i>rpoB</i>	<i>tonB</i>	without any MLST genes
actual class	<i>gapA</i>	1582 13.31 %	11 0.09 %	1 0.01 %	2 0.02 %	4 0.03 %	8 0.07 %	9 0.08 %	54 0.45 %
	<i>infB</i>	20 0.17 %	1462 12.30 %	4 0.03 %	6 0.05 %	11 0.09 %	11 0.09 %	2 0.02 %	54 0.45 %
	<i>mdh</i>	3 0.03 %	56 0.47 %	1370 11.53 %	42 0.35 %	45 0.38 %	7 0.06 %	28 0.24 %	93 0.78 %
	<i>pgi</i>	1 0.01 %	11 0.09 %	7 0.06 %	1454 12.23 %	2 0.02 %	43 0.36 %	4 0.03 %	137 1.15 %
	<i>phoE</i>	2 0.02 %	3 0.03 %	5 0.04 %	3 0.03 %	1098 9.24 %	3 0.03 %	5 0.04 %	41 0.34 %
	<i>rpoB</i>	1 0.01 %	1 0.01 %	1 0.01 %	17 0.14 %	4 0.03 %	1129 9.50 %	0 0.00 %	26 0.22 %
	<i>tonB</i>	0 0.00 %	0 0.00 %	9 0.08 %	2 0.02 %	0 0.00 %	4 0.03 %	955 8.03 %	34 0.29 %
	without any MLST genes	0 0.00 %	3 0.03 %	0 0.00 %	1 0.01 %	0 0.00 %	0 0.00 %	0 0.00 %	1996 16.79 %
		98.32 % 1.68 %	94.51 % 5.49 %	98.07 % 1.93 %	95.22 % 4.78 %	94.33 % 5.67 %	93.69 % 6.31 %	95.21 % 4.79 %	81.97 % 18.03 %
		92.93 % 7.07 %							

Figure 2 Confusion matrix for MLST loci classification results

Figure 3 Comparison of annotated gene signal length and gene signal length identified by the neural network, graph is shown without flyers for clearness

Figure 4 Example of multiple gene prediction
A. squiggle with labelled positions of gene (red) and random match (black), B. graph of predicted gene in squiggle, gene occurrence function is black, annotated position is red, C. multiple sequence alignment of *gapA* template, *gapA* hit and random match.

Figure 5 Architecture of NanoGeneNet

Each encoder block contains two convolution layers (Conv) with a kernel size equal to 3, followed by ReLU activation function and finally a Batch Normalization (BN) and a Maxpool layer with size kernel and stride of 2. Network also includes two Long Short-Term Memory (LSTM) recurrent networks and classification blocks composed of Fully Connected layers (FC), SoftMax activation layers and a DropOut layer used only in the training phase. For each block, tensor size shown in the form of *squiggle length x feature number*, where N is original squiggle length and M is its downsampled version length.

Supplementary material

Table S1 Number of squiggles containing given MLST locus for each analysed genome and number of squiggles with no MLST genes for given sequencing run

Table S2 Sequencing runs information