MinCall — MinION end2end deep learning basecaller

Neven Miculinić, Marko Ratković, and Mile Sikić
{neven.miculinic, marko.ratkovic, mile.sikic}@fer.hr

Faculty of Electrical Engineering and Computing (FER), Zagreb, Croatia

Abstract. The Oxford Nanopore Technologies's MinION is the first portable DNA sequencing device. It's capable of producing long reads, over 100 kBp were reported, however, it has significantly higher error rate than other methods.
In this study, we created MinCall, an end2end basecaller model for the MinION. The model is based on deep learning and uses convolutional neural networks (CNN) in its implementation. For extra performances is uses cutting edge deep learning techniques and architectures, batch normalization and Connectionist Temporal Classification (CTC) loss. The best performing deep learning model achieves state-of-the-art 91.4% median match rate on E.Coli dataset using R9 pore chemistry and 1D reads.

Keywords: Basecaller, MinION, R9, CNN, CTC, Next generation sequecing

1 Introduction

In recent years, deep learning methods significantly improved the state-of-the-art in multiple domains such as computer vision, speech recognition, and natural language processing [20] [18]. In this paper, we present application of deep learning in the field of Bioinformatics for DNA basecalling problem.

Oxford Nanopore Technology's MinION nanopore sequencing platform [25] is the first portable DNA sequencing device. It's small weight, of only 90 grams, low capital cost, and long read length combined with high-throughput, real-time data analysis, and decent accuracy yield promising results in various applications. From clinical application such as monitoring infectious disease outbreaks [15] [27], characterizing structural variants in cancer [26] and even full human genome assembly [14].

Although MinION is able to produce long reads, even up to 882 kb [23, 24], they have a high sequencing error rate. This has been somewhat alleviated with new R9 pore model, replacing older R7 ones. In this paper, we show that this error rate can be reduced by our approach with the properly trained neural network model.

The exact error rate metric is unreliable since multiple pipeline tools could be the issue. First the sample is prepared, hopefully, uncontaminated and matching reference genome as close as possible then sequenced using the MinION device obtaining raw data. Next, our model (or any other) comes along and basecalls the sequences. To evaluate error rate metric basecalled read is aligned to the reference genome using aligners with their own errors/biases, most commonly used BWA-MEM [21] and Graphmap [30].

2 Sequencing overview

Conceptually, the MinION sequencer works as follows. It's a variation on now standard shotgun sequencing approach. First, DNA is sheared into smaller DNA fragments and adapters are ligated to either end of the fragments. The resulting DNA fragments pass through a protein embedded in

a membrane via a nanometre-sized channel, a nanopore. A single DNA strand passes through the pore. Optionally, hairpin protein adapter can merge two DNA strands, allowing both template and complement read passing through the nanopore sequentially for more accurate reads. This technique is referred as 2D reads, while we focus on 1D reads containing only template DNA and no hairpin adapter.

As DNA strand passes through the nanopore, they are propelled by the current. However, this current varies depending on specific nucleotide context within the nanopore, changing its resistance. We sample the currency multiple times per second, 4000Hz in our dataset, and from this data, we deduce passing DNA fragment. By design, the nanopore is 6 nucleotides wide, and many models use this information internally, yet we opted ours of it in hopes of a simpler model, less feature engineering, and better homopolymer detection.

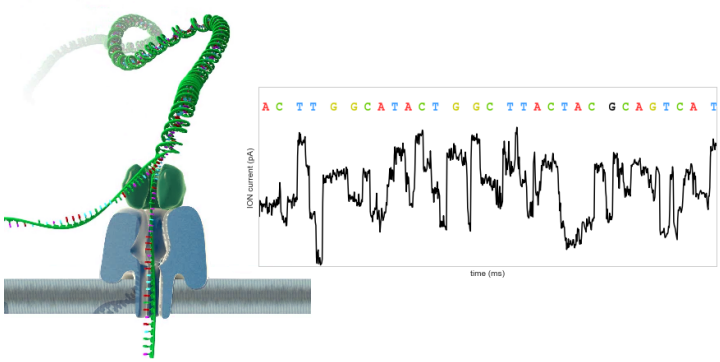

Fig. 1. DNA strain being pulled through a nanopore [1]

3 Basecalling

The core of the decoding process is the basecalling step, that is translating the current samples to the nucleotide sequence. Nowadays there are multiple basecalling options, what official and unofficial ones.

Earlier models were Hidden Markov model(HMM)-based where hidden state modeled DNA sequence of length 6 (6-mer) in the nanopore. Pore models were used in computing emission probabilities. [22, 28, 32, 33] and the recent open source HMM-based basecaller Nanocall [5]. Modern basecallers use RNN base models, and we opted out using CNN instead with beam search.

We compared our model on R9 chemistry with Metrichor (HMM-based approach), Nanonet and DeepNano (RNN based approaches). For the detailed basecaller overview see the appendix A.

[1] Figure adapted from https://nanoporetech.com/how-it-works

4 Dataset

Dataset used were E.Coli K-12 strands from [23] and Lamba basecalling[2]. Both used datasets show in table 1 have been previously passed through MinKNOW and had been basecalled by Metrichor. As 1D read analysis was the focus of this paper, only those reads were used.

Table 1. Used datasets

	Number of reads	Total bases [bp][3]	Whole genome size [bp]
E. Coli[4]	164471	1 481 687 490	4 639 675
lambda[5]	86	466 465	48 502

4.1 Data preprocessing

To help training process, the raw signal is split into smaller blocks that are used as inputs. For each Metrichor basecalled event is easy to determine the block it falls into using *start* field. Using this information output given by Metrichor can be determined for each block. To correct errors produced by Metrichor and possibly increase the quality of data, each read is aligned to the reference. This is done using aligner GraphMap [30] that returns the best position in the genome, hopefully, the part of the genome from which read came from. Alignment part in the genome is used as a target. Using CIGAR string returned by aligner we can correct Metrichor data and get target output for each block. This process is shown in figure 2.

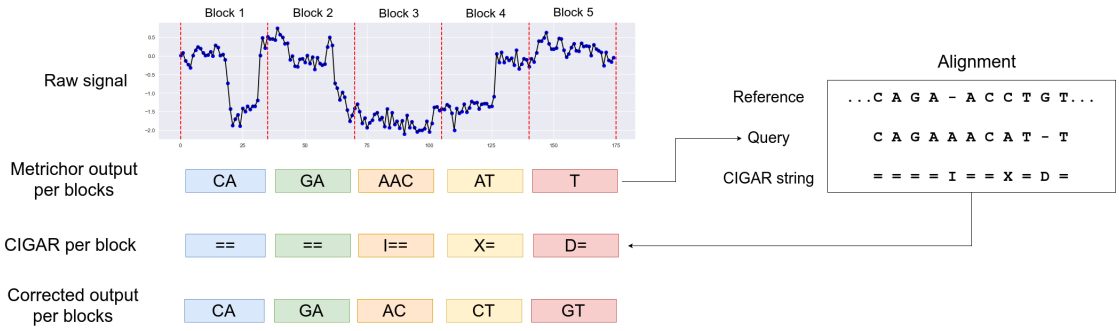

Fig. 2. Dataset preparation

[2] Acquired from doc. dr. sc. Petra Korać and dr.sc. Paula Dobrinić

[3] Total number of bases calle by Metrichor

[4] R9 sequencing data from `http://lab.loman.net/2016/07/30/nanopore-r9-data-release/`, reference taken from `https://www.ncbi.nlm.nih.gov/nuccore/48994873`

[5] Internal dataset acquired from doc. dr.sc. Petra Korać and dr.sc. Paula Dobrinić, reference taken from `https://www.ncbi.nlm.nih.gov/nuccore/NC_001416.1`

To eliminate the possibility of overfitting to the known reference, the model is trained and tested on reads from different organisms. Due to the limited amount of public available raw nanopore sequence data, E. Coli was *divided* into two regions. Reads were split into train and test portions, depending on which region of E. Coli they align. If read aligns inside first 70% of the E. Coli, it is placed into train set, and if it aligns to the second portion, it is placed into test set. Reads whose alignment overlaps train and test region are not used. Important to note that E. Coli genome, and genomes of the majority of other bacteria, is cyclical, so reads with alignments that wrap over edges are also discarded. Total train set consist of over 110 thousand reads. Overview of the entire learning pipeline is shown in figure 3.

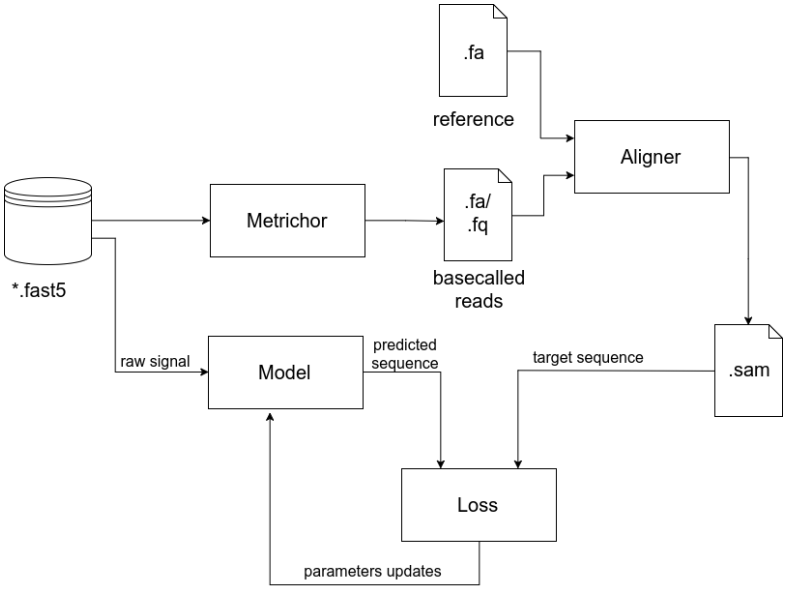

Fig. 3. Training pipeline overview

```
Target       :  A  G  A  A  A
Preprocessed:  A  G  A  A' A
```

Fig. 4. Target nucleotide sequence preprocessing

Due to CTC merged nature during decoding, adjacent duplicates are merged into one, we preprocess the target nucleotide sequence with surrogate nucleotides, such that each second repeated nucleotide is replaced with its surrogate. Example provided in figure 4. All raw input data were normalized to zero mean and unit variance as it yields superior performance with neural networks.

5 Method

Instead of opting for the traditional path using HMM or newly adopted RNN we tried using CNN (Convolutional neural networks) [19], that is their residual version [11]. For loss, we used CTC (Connectionist Temporal Classification) [9] between basecalled and the target sequence. Other building blocks used are Batch normalization(BN) [13] and pooling layers. Dropout [31] was not used. Described model is implemented using tensorflow [1] and source warp-ctc [2] GPU CTC loss implementation.

The final model is a residual neural network consisting of 72 residual blocks BN[6]-ELU[7]-CONV[8]-BN-ELU-CONV, to a grand total of 2 million parameters. The used model is a variant of architecture proposed in paper [12] with the difference of ELU being used as activation instead of ReLU as it is reported [29] to speeds up the learning process and improve accuracy as the depth increase.

Each convolutional layer in this models uses 64 channels with kernel size 3. Because sequenced read is always shorter than the raw signal, pooling with kernel size two is used every 24 layers resulting in a reduction of dimensionality by factor 8. This is beneficial in faster learning, better generalization and increased basecalling speed.

Training the model is the minimization of previously described CTC loss. It was done using Adam [16] with default parameters, and exponentially decaying learning rate starting from 1e-3 and decay rate of 5e-6 over 100k steps[9] and minibatch size 8. To prevent gradients exploding on bad inputs, they were clipped to a range [-2, 2]. We observed no overfitting due to large dataset size.

During testing, we tried ReLu and PrELU [10] with no significant result difference. We also tried different channel numbers, receptive field width, and various other hyperparameters during hyperparameter optimization and conclude after enough complexity, that is sufficient layers, all choices were performing similarly. We used SigOpt [7][10] bayesian hyperparameter optimization library.

6 Results

Developed tool was compared with other available basecallers that support R9 chemistry. This includes third-party basically DeepNano and official basecallers by Oxford Nanopore (cloud-based Metrichor and Nanonet). The fact that ground truth is not known makes evaluation difficult. Different methods for evaluation were used to get clearer information about each basecaller.

6.1 Per read metrics

A portion of the read length that aligns as correctly is called match_rate. Same goes for mismatches and insertions. Sum of all matches, mismatches, and insertions is equal to the reads length 1. For specific details see Appendix B.1. Results on E.Coli test set with Graphmap aligner

[6] Batch normalization

[7] Exponential Linear Unit

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0 \\ \alpha(exp(x) - 1), & \text{otherwise} \end{cases}$$

[8] 1D convolutional layer

[9] We use tf.train.exponential_decay where current learning rate, lr is $lr = \text{initial\_lr} \cdot \text{decay\_rate}^{\frac{\text{global\_step}}{\text{decay\_step}}}$

[10] `https://sigopt.com/`

are shown in table 2. Evaluation with BWM-MEM and on lambda can be found in appendix B.1 under tables 5, 6 and 7. Furthermore we plot Kernel Density estimation(KDE) plots for each mentioned statistic on E.Coli dataset in figure 5.

In Appendix B.1 the histogram figure 7 shows how matches, mismatches, insertions, and deletions are distributed across the reads. It is shown that mismatches and insertion occur more frequently at the beginnings and the ends of the reads. This is not only the case for the developed basecaller, but all other shows the same property. This could be due to lack of context information from both sides when edges are basecalled.

Table 2. Alignment specifications of E. Coli R9 basecalled reads using GraphMap

	Match % (median)	Mismatch % (median)	Insertion % (median)	Deletion % (median)
DeepNano	90.254762	6.452852	**3.274420**	11.829965
Metrichor	90.560455	5.688105	3.660381	8.328271
Nanonet	90.607674	5.608912	3.652791	8.299046
MinCall	**91.408591**	**5.019141**	3.477739	**7.471608**

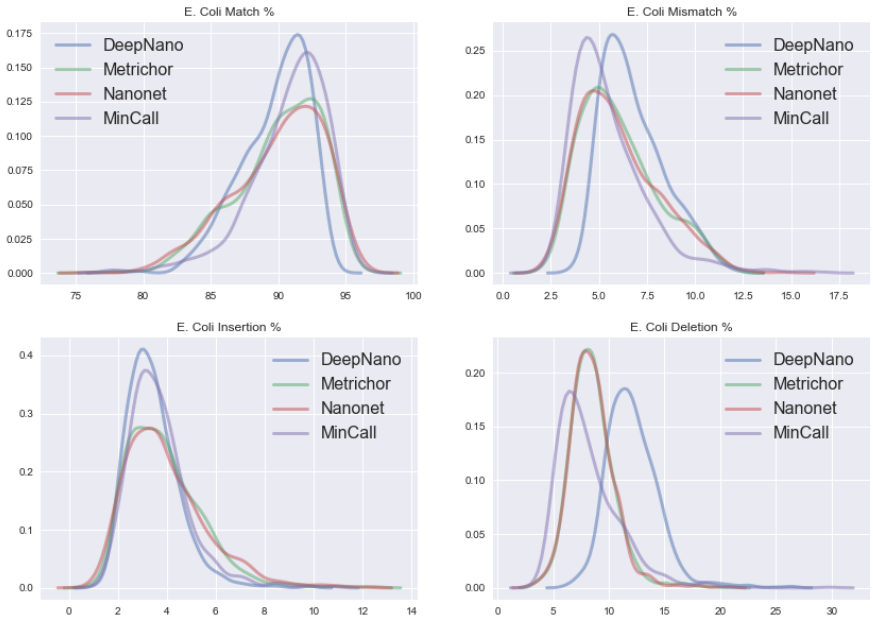

Fig. 5. KDE plot for distribution of percentage of alimnment operations for E. Coli

6.2 Consensus metrics

In Subsection 6.1 we showed metrics after the read is aligned to reference genome. However, if basecalled simply yield nucleotide sequence A irrespective of ground truth, it shall have 100% match rate since it will definitely align somewhere. Therefore we reconstruct the original genome from basecalled reads. Specific metrics definitions analysed are in Appendix B.2.

Consensus from pileup Instead of going through the whole assembly process, as we know the reference genome of the data used in these tests, we simply align all the reads to the genome, stack them on top of each other forming pileup of read bases. Using majority vote, dominant bases are called on each position. The resulting sequence is called consensus. When calling consensus for deletions, there has to be a majority of deletions of the same length. Calling insertions has the additional condition, the majority has to agree on both length and the bases of insertion. Figure 6 shows how consensus is called from pileup created from aligned reads. Pileup is stored in mpileup format.

All models show a slight bias towards deletions than insertions, but this may be the limitation of technology as it has been reported that deletion and mismatch rates for nanopore data are ordinarily higher than insertion rates [30]. Results are shown in table 3. Lambda table is deferred to Appendix in table 8.

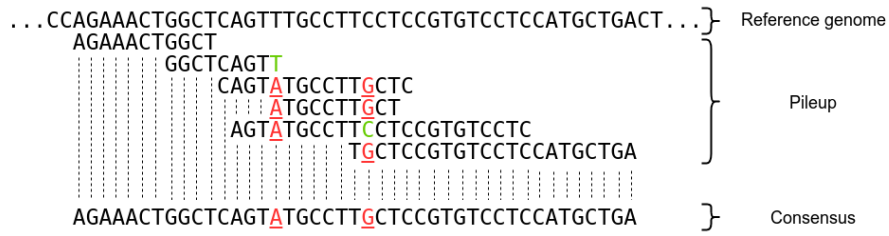

Fig. 6. Consensus from pileup

Table 3. Consensus specifications of E. Coli R9 basecalled reads

	Total called [bp]	Correctly called [bp]	Match %	Snp %	Insertion %	Deletion %
DeepNano	1510244.0	1493242.0	98.8742	1.0044	0.1214	0.9041
Metrichor	1515893.0	1502588.0	99.1223	0.7464	0.1313	0.6300
Nanonet	1414237.0	1385515.0	97.9691	1.5700	0.4609	1.5158
resdeep	1517828.0	1506233.0	**99.2361**	**0.6474**	**0.1165**	**0.5510**

Consensus from de novo assembly In this evaluation method, the consensus sequence is not calculated from the pileup, but by de novo genome assembly. For this task, fast and accurate de novo genome assembler *ra*[11] [34] was used and obtained consensus sequence is compared to the reference using dnadiff present in the Mumer[12]. The length of the reference, consensus sequence, number of contigs and percentages of aligned bases from the reference to the query and vice versa are shown in the table 4. Average identity summarizes how closely does the assembled sequence match the reference. This is run on full E. Coli sequence run for 1D template reads (˜160k reads), for both developed tools, Nanonet and Metrichor. Developed tool has shown a small increase in quality of the assembled sequence over Metrichor by offering longer consensus, higher identity percentage, and overall smaller edit distance[13].

Table 4. Assembly and consensus results for E. Coli

	Metrichor	MinCall	Nanonet
Ref. genome size (bp)	4639675	4639675	4639675
Total bases (bp)	4604806	**4614354**	4600056
Contigs [#]	1	1	1
Aln. bases ref. (bp)	4639641(100.00%)	4639612(100.00%)	4639031(99.99%)
Aln. bases query (bp)	4604787(100.00%)	4614351(100.00%)	4599745(99.99%)
Avg. Identity	98.76	**99.06**	98.47
Edit distance	60418	**46686**	74341

7 Conclusion and further work

In this paper, we used CNN instead of already tried RNN or HMM approaches, which resulted in in higher accuracy compared to other existing basecallers. Unlike HMM and RNN, there's no explicit dependency on previous hidden state, therefore this model is massively parallelizable and more sensible given data nature — that is we're dealing with signal processing, not heavily context-depended language modeling.

All test are done on data for R9 chemistry, but the developed open source code could easily be adjusted and trained on R9.4 and newest R9.5 data when it becomes publicly available.

Currently, without support for newer sequencing data, this model has limited application. It can be used as a demonstration of a different approach to basecalling which yields promising results. As newer versions of basecallers by Oxford Nanopore do not offer any support for data sequenced with the previous version of chemistries, this tool can be used to re-basecall that data and improvement of the quality of reads retrospectively.

Future work includes experiments with recently proposed Scaled exponential linear units (SELU) [17] that eliminate the need for normalization techniques such as used batch normalization. Possible improvements of the model include the combination of convolutions and attention

[11] `https://github.com/rvaser/ra`
[12] `https://github.com/garviz/MUMmer`
[13] Calculated using `https://github.com/isovic/racon/blob/master/scripts/edcontigs.py`

mechanism proposed just recently in the paper [8] showing excellent results in both speed and accuracy, for tasks of language translation. The other option could usage of stacked simple models, such as logistic regression and SVM to predict each nucleotide given raw signal context, similar to our deep learning model pre-CTC layer, and use linear chain CRF for full sequence basecalling.

8 Acknowledgments

During this paper creation, we used Sigopt academic license for hyperparameter optimization. We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research.

We're thankful to various other people whose code, tools and advice we've used in completing this project: Fran Jurišić, Ana Marija Selak, Ivan Sović, Robert Vaser and Martin Šošić.

For data, we're thankful for the cooperation of Biologists doc. dr. sc. Petra Korać and dr.sc. Paula Dobrinić and Loman labs [23] for publicly posting their dataset.

References

1. Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G.S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., Zheng, X.: TensorFlow: Large-scale machine learning on heterogeneous systems (2015), `http://tensorflow.org/`, software available from tensorflow.org
2. Amodei, D., Anubhai, R., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Chen, J., Chrzanowski, M., Coates, A., Diamos, G., Elsen, E., Engel, J., Fan, L., Fougner, C., Han, T., Hannun, A.Y., Jun, B., LeGresley, P., Lin, L., Narang, S., Ng, A.Y., Ozair, S., Prenger, R., Raiman, J., Satheesh, S., Seetapun, D., Sengupta, S., Wang, Y., Wang, Z., Wang, C., Xiao, B., Yogatama, D., Zhan, J., Zhu, Z.: Deep speech 2: End-to-end speech recognition in english and mandarin. CoRR abs/1512.02595 (2015), `http://arxiv.org/abs/1512.02595`
3. Boža, V., Brejová, B., Vinař, T.: DeepNano: Deep recurrent neural networks for base calling in MinION nanopore reads. PLOS ONE 12(6), e0178751 (jun 2017), `https://doi.org/10.1371/journal.pone.0178751`
4. Community, N.: Basecalling overview, `https://community.nanoporetech.com/technical_documents/data-analysis/v/datd_5000_v1_reve_22aug2016/basecalling-overvi`, [Accessed; 12-July-2017]
5. David, M., Dursi, L.J., Yao, D., Boutros, P.C., Simpson, J.T.: Nanocall: an open source basecaller for oxford nanopore sequencing data. Bioinformatics p. btw569 (2016)
6. David, M., Dursi, L.J., Yao, D., Boutros, P.C., Simpson, J.T.: Nanocall: An open source basecaller for oxford nanopore sequencing data. bioRxiv (2016), `http://biorxiv.org/content/early/2016/03/28/046086`
7. Dewancker, I., McCourt, M., Clark, S., Hayes, P., Johnson, A., Ke, G.: A strategy for ranking optimization methods using multiple criteria. In: Workshop on Automatic Machine Learning. pp. 11–20 (2016)
8. Gehring, J., Auli, M., Grangier, D., Yarats, D., Dauphin, Y.N.: Convolutional sequence to sequence learning (2017)
9. Graves, A., Fernández, S., Gomez, F., Schmidhuber, J.: Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In: Proceedings of the 23rd international conference on Machine learning. pp. 369–376. ACM (2006)

10. He, K., Zhang, X., Ren, S., Sun, J.: Delving deep into rectifiers: Surpassing human-level performance on imagenet classification (2015)
11. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 770–778 (2016)
12. He, K., Zhang, X., Ren, S., Sun, J.: Identity mappings in deep residual networks (2016)
13. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift (2015)
14. Jain, M., Koren, S., Quick, J., Rand, A.C., Sasani, T.A., Tyson, J.R., Beggs, A.D., Dilthey, A.T., Fiddes, I.T., Malla, S., et al.: Nanopore sequencing and assembly of a human genome with ultra-long reads. bioRxiv p. 128835 (2017)
15. Judge, K., Harris, S.R., Reuter, S., Parkhill, J., Peacock, S.J.: Early insights into the potential of the oxford nanopore minion for the detection of antimicrobial resistance genes. Journal of Antimicrobial Chemotherapy 70(10), 2775–2778 (2015)
16. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization (2014)
17. Klambauer, G., Unterthiner, T., Mayr, A., Hochreiter, S.: Self-normalizing neural networks (2017)
18. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Pereira, F., Burges, C.J.C., Bottou, L., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems 25, pp. 1097–1105. Curran Associates, Inc. (2012), https://goo.gl/UpFBv8
19. LeCun, Y., Bottou, L., Bengio, Y., Haffner, P.: Gradient-based learning applied to document recognition. Proceedings of the IEEE 86(11), 2278–2324 (November 1998)
20. LeCun, Y., Bengio, Y.: The handbook of brain theory and neural networks. chap. Convolutional Networks for Images, Speech, and Time Series, pp. 255–258. MIT Press, Cambridge, MA, USA (1998), http://dl.acm.org/citation.cfm?id=303568.303704
21. Li, H.: Aligning sequence reads, clone sequences and assembly contigs with bwa-mem. arXiv preprint arXiv:1303.3997 (2013)
22. Loman, N.J., Quick, J., Simpson, J.T.: A complete bacterial genome assembled de novo using only nanopore sequencing data. Nature methods 12(8), 733–735 (2015)
23. Loman, N.: Nanopore R9 rapid run data release, http://lab.loman.net/2016/07/30/nanopore-r9-data-release/, [Online; posted 30-July-2016]
24. Loman, N.: Thar she blows! Ultra long read method for nanopore sequencing, http://lab.loman.net/2017/03/09/ultrareads-for-nanopore/, [Online; posted 9-March-2017]
25. Mikheyev, A.S., Tin, M.M.: A first look at the oxford nanopore minion sequencer. Molecular ecology resources 14(6), 1097–1102 (2014)
26. Norris, A.L., Workman, R.E., Fan, Y., Eshleman, J.R., Timp, W.: Nanopore sequencing detects structural variants in cancer. Cancer biology & therapy 17(3), 246–253 (2016)
27. Quick, J., Loman, N.J., Duraffour, S., Simpson, J.T., Severi, E., Cowley, L., Bore, J.A., Koundouno, R., Dudas, G., Mikhail, A., et al.: Real-time, portable genome sequencing for ebola surveillance. Nature 530(7589), 228–232 (2016)
28. Schreiber, J., Karplus, K.: Analysis of nanopore data using hidden markov models. Bioinformatics p. btv046 (2015)
29. Shah, A., Kadam, E., Shah, H., Shinde, S., Shingade, S.: Deep residual networks with exponential linear unit (2016)
30. Sović, I., Šikić, M., Wilm, A., Fenlon, S.N., Chen, S., Nagarajan, N.: Fast and sensitive mapping of nanopore sequencing reads with graphmap. Nature communications 7 (2016)
31. Srivastava, N., Hinton, G.E., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1), 1929–1958 (2014)
32. Szalay, T., Golovchenko, J.A.: De novo sequencing and variant calling with nanopores using poreseq. Nature biotechnology 33(10), 1087–1091 (2015)
33. Timp, W., Comer, J., Aksimentiev, A.: Dna base-calling from a nanopore using a viterbi algorithm. Biophysical journal 102(10), L37–L39 (2012)
34. Vaser, R., Sovic, I., Nagarajan, N., Sikic, M.: Fast and accurate de novo genome assembly from long uncorrected reads. bioRxiv (2016), http://biorxiv.org/content/early/2016/08/05/068122

A Basecallers

Here is currated basecaller list:

A.1 Official

Metrichor is an Oxford Nanopore company that offers cloud-based platform *EPI2ME* for analysis of nanopore data. Initially, basecalling was only available by uploading data to the platform - that being the reason why this basecaller is often called Metrichor even though it is a name of the company.

With the release of R9 chemistry, this model was replaced by a more accurate recurrent neural network (RNN) implementation. Currently, Oxford Nanopore offers several RNN-based local basecaller versions under different names: Albacore, Nanonet, and basecaller integrated into MinKNOW [4].

Albacore is basecaller by Oxford Nanopore Technologies ready for production and actively supported. It is available to the Nanopore Community served as a binary. The source code of Albacore was not provided and is only available through the ONT Developer Channel. The tool supports only R9.4 and future R9.5 version of the chemistry. For R9 tests in this paper, we used R9.4 chemistry setting as instructed on ONT forums.

Nanonet[14] uses the same neural network that is used in Albacore but it is continually under development and does contain features such as error handling or logging needed for production use. It uses the *CURRENNT* library for running neural networks. It supports basecalling of both R9 and R9.4 chemistry versions. However, in our experiments it was painfully slow, which was as expected due to it's classification as not production ready.

Scrappie[15] is another basecaller by Oxford Nanopore Technologies. Similar to Nanonet, it is the platform for ongoing development. Scrappie is reported to be the first basecaller that specifically address homopolymer basecalling. It became publicly available just recently in June 2017 and supports R9.4 and future R9.5 data.

A.2 Third-party basecallers

Nanocall [6] was the first third-party open source basecaller for nanopore data. It uses HMM approach like the original R7 Metrichor. Nanocall does not support newer chemistries after R7.3.

DeepNano [3] was the first open-source basecaller based on neural networks. It uses bidirectional recurrent neural networks implemented in Python, using the Theano library. When released, originally only supported R7 chemistry, but support for R9 and R9.4 was added recently.

B Evaluation metrics

B.1 CIGAR derived metrics

A portion of the read length that aligns as correctly is called match_rate. Same goes for mismatches and insertions. Sum of all matches, mismatches, and insertions is equal to the reads length 1. Here is the specific equation list:

$$read\_len = n\_matches + n\_mismatches + n\_insertions \tag{1}$$

[14] `https://github.com/nanoporetech/nanonet/`
[15] `https://github.com/nanoporetech/scrappie`

$$match\_rate = \frac{n\_matches}{read\_length} \tag{2}$$

$$missmatch\_rate = \frac{n\_mismatches}{read\_length} \tag{3}$$

$$insertion\_rate = \frac{n\_insertions}{read\_length} \tag{4}$$

$$match\_rate + snp\_rate + insertion\_rate = 1 \tag{5}$$

Deletion rate is defined as a total number of deletions in the alignment over the length of the aligned read.

$$deletion\_rate = \frac{n\_deletion}{read\_length} \tag{6}$$

Table 5. Alignment specifications of E. Coli R9 basecalled reads using BWA mem

	Match % (median)	Mismatch % (median)	Insertion % (median)	Deletion % (median)
DeepNano	90.254762	6.452852	3.274420	11.829965
Metrichor	90.595441	6.869543	2.531646	7.567381
Nanonet	90.988989	6.674760	**2.348552**	7.698530
MinCall	**91.470588**	**5.929204**	2.477283	**6.970362**

Table 6. Alignment specifications of Lambda basecalled reads using GraphMap

	Match % (median)	Mismatch % (median)	Insertion % (median)	Deletion % (median)
DeepNano	86.997687	9.623494	3.442490	16.052830
Metrichor	87.714988	7.835052	4.093851	**10.757491**
Nanonet	88.415611	8.178372	3.629653	11.793022
MinCall	**89.694482**	**7.238095**	**3.078796**	13.450292

B.2 Consensus metrics

Similarly, as before, match rate, mismatch rate, insertion and deletion rates are calculated but this time for whole consensus sequence. In this context, mismatches are called *single nucleotide polymorphisms* (snp).

$$match\_rate = \frac{n\_correct\_bases}{consensus\_length} \tag{7}$$

$$snp\_rate = \frac{n\_snp}{consensus\_length} \tag{8}$$

Table 7. Alignment specifications of lambda R9 basecalled reads using BWA mem

	Match % (median)	Mismatch % (median)	Insertion % (median)	Deletion % (median)
DeepNano	86.625973	11.288361	2.098225	14.648308
Metrichor	87.294093	10.109186	2.376476	**9.645323**
Nanonet	87.767037	10.017598	2.354248	10.597232
MinCall	**89.049870**	**9.480883**	**1.615188**	12.962441

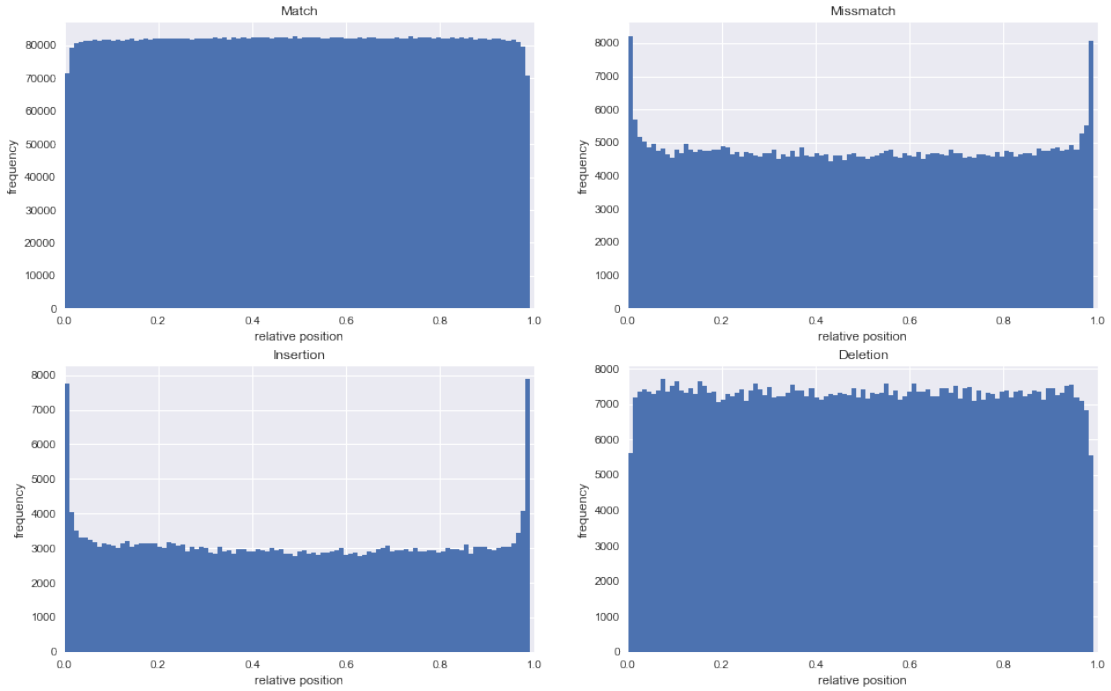

Fig. 7. Histogram of alignment operations over relative position inside of read

$$insertion\_rate = \frac{n\_insertions}{consensus\_length} \tag{9}$$

$$match\_rate + snp\_rate + insertion\_rate = 1 \tag{10}$$

$$deletion\_rate = \frac{n\_deletion}{consensus\_length} \tag{11}$$

Table 8. Consensus specifications of lambda R9 basecalled reads

	Total called [bp]	Correctly called [bp]	Match %	Snp %	Insertion %	Deletion %
DeepNano	48342.0	48025.0	99.3443	0.6433	**0.0124**	0.2648
Metrichor	48469.0	48257.0	**99.5626**	**0.4188**	0.0186	**0.1465**
Nanonet	48438.0	48168.0	99.4426	0.5409	0.0165	0.1961
resdeep	48385.0	48163.0	99.5412	0.4402	0.0186	0.1976