Deepbinner

Demultiplexing barcoded Oxford Nanopore reads with deep convolutional neural networks

Ryan R. Wick[1*], Louise M. Judd[1], Kathryn E. Holt[1]

1 Department of Biochemistry and Molecular Biology, Bio21 Molecular Science and Biotechnology Institute, University of Melbourne, Parkville, Victoria 3010, Australia

* rrwick@gmail.com

Abstract

Multiplexing, the simultaneous sequencing of multiple barcoded DNA samples on a single flow cell, has made Oxford Nanopore sequencing cost-effective for small genomes. However, it depends on the ability to sort the resulting sequencing reads by barcode, and current demultiplexing tools fail to classify many reads. Here we present Deepbinner, a tool for Oxford Nanopore demultiplexing that uses a deep neural network trained on 9.1 GB of data, to classify reads based on the raw electrical read signal. This 'signal-space' approach allows for greater accuracy than existing 'base-space' tools (Albacore and Porechop) in which signals have first been converted to DNA base calls, itself a complex problem that can introduce noise into the barcode sequence. To assess Deepbinner and existing tools, we performed multiplex sequencing on 12 amplicons chosen for their distinguishability. This allowed us to establish a ground truth classification for each read based on internal sequence alone. Deepbinner had the lowest rate of unclassified reads (5.2%) and the highest demultiplexing precision (98.4% of classified reads were correctly assigned). It can be used alone (to maximise the number of classified reads) or in conjunction with Albacore (to maximise precision and minimise false positive classifications). We also found cross-sample chimeric reads (0.3%) and evidence of barcode switching (1%) in our dataset, which likely arise during library preparation and may be detrimental for quantitative studies that use multiplexing. Deepbinner is open source (GPLv3) and available at https://github.com/rrwick/Deepbinner.

Introduction

Oxford Nanopore barcoding

Multiplexing (barcoding) is a common strategy used to distribute high-throughput DNA sequencing capacity over multiple samples [1]. For each input DNA sample, a unique barcode is incorporated into the library of DNA molecules prepared for sequencing. Multiple barcoded DNA libraries can then be combined and sequenced simultaneously on the same flow cell. The resulting reads must then be demultiplexed: sorted into bins according to the barcode sequence. Barcoding has obvious economic advantages, allowing users to divide the fixed cost of a sequencer flow cell over multiple input samples.

When the Oxford Nanopore Technologies (ONT) MinION sequencer was first released, its yield was measured in hundreds of Megabases (Mbp) and effective sequencing of a bacterial genome required an entire flow cell [2]. The last four years have seen a nearly 100-fold increase in yield, with 10 Gbp or more now possible from a MinION sequencing run [3]. ONT's native barcoding kit for 1D ligation sequencing (EXP-NBD103) provides 12 barcodes which are ligated onto both ends of the DNA molecules to be sequenced. This kit allows the sequencing capacity of a single MinION run to be distributed across 12 bacterial genomes which can thus be simultaneously sequenced on a single flow cell [4].

Each ONT sequencing read is generated as an electrical signal composed of variations in electrical current as the DNA molecule moves through the nanopore. The MinION sequencer measures the current at 4 kHz and the DNA advances at a rate of 450 bases/sec, equating to 8.9 current measurements per base, on average. These 'signal-space' reads (a.k.a. raw signal) are translated into 'base-space' nucleotide sequences by basecalling software [5–7]. Basecalling is an inexact process and the resulting reads have a per-base error rate of 5–25% [8]. This error rate can be a problem for downstream analyses, including current ONT barcode demultiplexing tools, such as Albacore and Porechop, where it is common for up to 20% of the barcoded reads to be unassigned to a bin and therefore unusable [4]. Other types of ONT read analyses often achieve better performance by working with the raw signal instead [9, 10].

Convolutional neural networks

In the last decade, neural networks – specifically convolutional neural networks (CNNs) – have revolutionised the field of image classification, achieving record high accuracies [11, 12]. This progress has been fuelled by general-purpose computing on graphics processing units (GPUs) which allow much faster performance when training and classifying with CNNs, and have in turn allowed for deeper and more complex CNNs than were previously feasible. Despite their impressive accuracy, deep CNNs have been criticised for their incomprehensibility – it can be difficult to tell how or why a CNN classifier made a particular decision [13].

Barcode classification using ONT raw signal is conceptually similar to image classification, but it is a simpler problem in two key aspects. First, ONT raw signal is a one-dimensional array of values whereas images typically have three dimensions (height, width and channels). Second, there are a smaller number of possible barcode classes (12 to 96, depending on the kit used) than possible image classes (often 1000 or more) [14].

Deepbinner

Here we present Deepbinner, a tool for ONT barcode demultiplexing using a deep CNN, to classify reads into barcode bins using the raw read signal. We compare its performance with that of other ONT demultiplexing tools, Albacore and Porechop, which work in base-space. Operating in signal-space gives Deepbinner more power to demultiplex reads and the ability to sort raw reads for downstream uses such as Nanopolish [9]. By taking a machine learning approach to demultiplexing, Deepbinner relies on no prior knowledge of barcode sequences or signals – it learns how to classify reads from a labelled training set. Here we demonstrate its use with the EXP-NBD103 set of 12 barcodes, but it could equally be trained on any barcode set.

Design and implementation

Deepbinner

Network architecture

Deepbinner is implemented as a deep CNN using the TensorFlow [15] and Keras [16] code libraries. Its neural network architecture was based on elements developed in the field of image classification: groups of convolutional layers followed by max pooling layers [17]; parallel 'inception' modules and low dimension bottlenecks [18]; noise, dropout and batch normalisation layers [19]; and global average pooling [20].

Using these elements, we trialled hundreds of randomised network architectures to search for an effective design. Networks were assessed on their loss (categorical cross-entropy) and classification accuracy on a validation set. To discourage overfitting, we preferred models with fewer parameters and a small validation set loss to training set loss ratio. The best performing architecture was subsequently refined to produce the final Deepbinner network shown in Fig 1. One notable way that Deepbinner's architecture differs from image classification networks is the number of filters. Deepbinner uses a constant filter size (48, except for where the parallel module increases the filter count) whereas image classification networks commonly use a smaller number of filters in early layers and a larger number of filters in later layers.

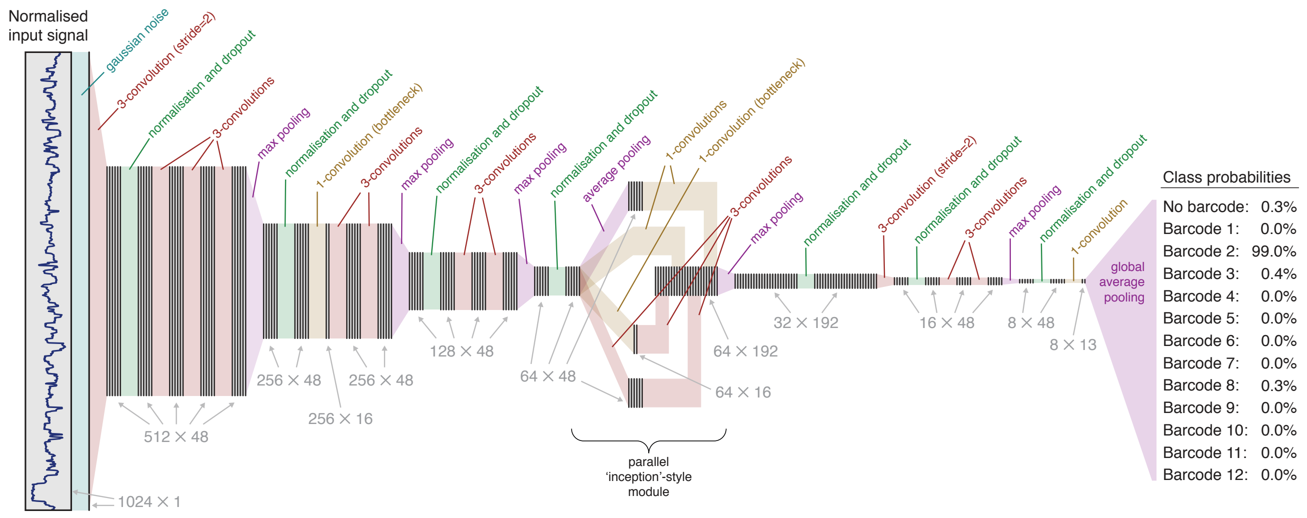

Figure 1. Neural network architecture. Layers in the network are drawn as coloured blocks and data as groups of vertical lines. Data dimensions are shown for each step of the process as *data length* × *filter count*. Gaussian noise and dropout layers are only active during network training, not during classification.

Deepbinner's input array size of 1024 was chosen two reasons. First, it is a power of two, allowing the data to halve in length as it progresses through the network's dimension-reducing layers. Second, it is sufficiently long to capture an entire ONT 24 bp barcode, which has an expected length of 213 electrical current values.

Network training

Our training data came from eight R9.4 flow cells and six R9.5 flow cells, all used with the EXP-NBD103 barcoding kit. We basecalled the reads using Albacore (v2.2.7) and then demultiplexed them with Porechop (v0.2.3) using stringent settings to select reads with matching barcodes on both the start and end of the read. Both start-read signal (raw signal from the beginning of the read) and end-read signal (raw signal from the end of the read) were extracted from each fast5 file and any open-pore signals (high current values corresponding to the absence of a DNA molecule in the nanopore [5]) were trimmed off. This produced a total of 2,190,485 training samples, each an array of 1024 electrical current values with a corresponding barcode label. We then balanced the training data (randomly subsetting each barcode's samples down to the number of samples for the least abundant barcode) and removed samples of insufficient length, leaving 803,665 read-start samples and 803,294 read-end samples.

In addition to training Deepbinner on barcoded signals, we also included a variety of signals in the training set without a barcode that were assigned a corresponding no-barcode class. These included signals from real sequences that lacked barcodes (taken from the middle of reads) and multiple types of simulated signals: flat signal, Gaussian noise and Perlin noise (Fig S1). Their presence in the training set ensures that Deepbinner can actively assign reads to a no-barcode class, not just fail to find a strong match. These no-barcode samples were added at a rate to make up $\frac{1}{3}$ of the training samples, bringing the training set sizes to 1,207,231 and 1,206,860 for read-starts and read-ends.

Data augmentation is a method of artificially expanding a training set by duplicating samples with transformations [21]. Deepbinner applies data augmentation by distorting signals in the temporal dimension (Fig S2). This is carried out by duplicating current values at $\frac{1}{4}$ of the signal's positions (randomly chosen) and deleting values at $\frac{1}{4}$ of the positions. The result is a signal of equal length but elongated in some places and shortened in others. We did not implement distortion of the signal amplitude because a similar role is carried out by the network's Gaussian noise layer during training (Fig 1). For our dataset, we found that an augmentation factor of two (one augmented signal for each unmodified signal) produced an ideal balance between training time and classification accuracy. However, this factor is adjustable, and datasets with fewer training samples may benefit from more data augmentation.

We refined our training dataset by repeatedly training a model, using it to classify the training data and discarding samples that the network struggled to classify. This served to remove samples which did not contain a barcode signal because the open-pore signal was improperly trimmed. This removed approximately 1% of the samples, leaving final training sets of 1,181,431 and 1,200,132 samples for the starts and ends of reads, respectively (totalling 9.1 GB in size).

The model's accuracy was assessed using an 80:20 training:validation split of the training data (Fig S3). Data augmentation was only performed on the training data which, combined with the network's Gaussian noise and dropout layers (only active during training), explains why validation loss and accuracy was superior to training loss and accuracy.

We produced the final Deepbinner trained models by training the network for 100 epochs using these entire sets (i.e. no training-validation split) and a data augmentation factor of two. Each model was trained on a single Tesla P100 GPU and took approximately 23 hours to complete.

Read classification

When classifying raw ONT signal, Deepbinner will generate a probability for each possible barcode, along with a no-barcode probability (Fig 1). Deepbinner will assign a barcode to the signal if the highest probability is sufficiently larger than the second-highest probability (the default minimum difference is 0.5, though this can be adjusted for more or less stringent binning). If the best and second-best matches are too close or if the best-match is 'no barcode', then Deepbinner assigns the 'none' label to the signal.

Deepbinner can classify each read using the start-read signal, the end-read signal or both. Using both start and end is appropriate for library preparations which add barcodes to both sides of a reads, such as EXP-NBD103. Other preparations, such as the SQK-RBK004 rapid barcoding kit, only add a barcode to the start of the read, in which case Deepbinner's start-only classification would be appropriate. If both start and end are used, Deepbinner will independently perform classification using the read start and read end. Reads will be binned if a sufficient match was found on either end, but if the two ends match different barcodes, the read will be considered a chimera and put in the 'none' bin. These reads can be identified by running Deepbinner in verbose mode which provides detailed information on barcode calls for both the start and end of reads. Deepbinner can optionally require a positive match on both the start and end to bin a read, enabling extra-stringent demultiplexing.

An ONT barcode signal usually appears in the first 1024 values of a read's raw signal, but this may not be the case for reads which contain a longer-than-normal amount of open-pore signal. Deepbinner therefore examines multiple 1024 signal windows overlapping by 512 samples. By default, it examines 11 such windows (covering 6144 samples in total), but this can be configured. Deepbinner keeps the results from the window which has the strongest probability for a bin other than 'none'.

Evaluation

Study design

In order to assess the accuracy with which Deepbinner can assign ONT reads to bins based on barcode signals, we aimed to sequence a library of barcoded DNA molecules for which the input source of each molecule could be verified independently of the barcode itself. To do this, we chose 12 bacterial isolates of different species that had been sequenced via Illumina HiSeq, which produces high accuracy short reads. To identify a region from each that was entirely unique to that genome, we performed a co-assembly of the pooled Illumina reads for all samples with SPAdes (v3.11.1) using a small k-mer setting (k=23). The longest contigs from this assembly were matched to their source genome by comparison to individual (non-pooled) genome assemblies, and we choose a single contig per source genome. This produced 12 sequences, one from each genome, each composed entirely of unique 23-mers, making them easily distinguishable from each other.

Amplicon library preparation and MinION sequencing

The 12 bacterial isolates were grown overnight at 37°C on LB agar plates. Single colonies were then grown overnight at 37°C in Luria broth. Bacterial cell pellets from 1.5 ml of broth culture were generated by centrifugation at 15,000×g for 5 minutes. DNA was extracted from these pellets using Agencourt GenFind v2 (Beckman Coulter) with minor modifications as follows. Cell pellets were resuspended in 400 µl lysis buffer containing 9 µl Proteinase K (96 mg/ml Beckman Coulter) and 1 µl RNase A (100 mg/ml Sigma Aldrich R6513) by gentle tip mixing. Samples were lysed at 37°C for 30 minutes. gDNA was extracted from the lysed samples by completing the remaining steps of the GenFind v2 for 200 µl of blood/serum from the binding step onwards.

We used the Primer3web tool (v4.1.0) to choose 25 bp long-range PCR primers from each of the 12 unique sequences to define amplicons which ranged from 9–11 kbp (File S2). PCR was performed using LongAMP Taq 2X Master Mix (New England Biolabs) with 150–450 ng gDNA as input template, primers at 1uM, an annealing temperature of 56°C and 35 cycles of amplification. Following the PCR, size selection purification was performed with Agencourt AMPure beads (Beckman Coulter) at 0.6× ratio. The size and specificity of the PCR product was confirmed by capillary electrophoresis (Fragment Analyser AATI). A sequencing library was prepared from the purified amplicons using the Nanopore 1D ligation sequencing kit (SQK-LSK108) with the native barcoding expansion kit (EXP-NBD103) as per the manufacturer's instructions. The run was performed on a MinION MK1b device using MinKNOW v18.03.1 and the `NC_48Hr_Sequencing_Run_FLO-MIN106_SQK-LSK108` protocol.

Demultiplexing

The amplicon reads and negative control reads were basecalled with Albacore (2.2.7) using the following options: `barcoding` (to enabled demultiplexing), `disable_filtering` (to include low-quality reads) and `basecaller.max_events=10000` (to improve basecalling quality). The resulting FASTQ files were pooled and shuffled, and then given to Porechop (v0.2.3) to be independently demultiplexed. The pre-basecalled raw fast5 files were demultiplexed with Deepbinner (v0.1.0). Both Porechop and Deepbinner were used with default settings.

We assigned ground truth classifications to the ONT reads by aligning their basecalled sequences to the amplicon reference sequences with minimap2 (v2.10) [22] and binning with the `assign_reads_to_reference.py` script (included with Deepbinner). Reads which failed to meet an alignment threshold (100 bp or 10% of the read length, whichever is smaller) were classified as 'unknown'. Reads which exceeded an alignment threshold to a secondary amplicon (50 bp or 5% of the read length, whichever is larger) were classified as 'chimera'. This method is only able to detect cross-bin chimeras – reads which separate components from two separate amplicons. It cannot not detect within-bin chimeras, e.g. two copies of the same amplicon concatenated in a single read. We also assessed classification using a negative control set of ONT reads (SQK-LSK108, R9.4 flow cell) that were not barcoded. The ground truth classification for all reads in the negative control set was 'none'.

Results

Performance on evaluation set

The barcoded amplicon MinION sequencing run produced 1,893,881 reads (9.69 Gbp), 1,646,884 of which (87%) could be reliably assigned to an amplicon based on the internal read sequence alone, producing our classification ground truth set. Our negative control set contained 218,148 reads (2.05 Gbp).

Of all three demultiplexers tested, Deepbinner performed best on both precision (positive predictive value) and recall (proportion of reads binned correctly) (Table 1 and File S1). It was particularly strong on recall, binning 349,818 more reads (1418 Mbp) than Albacore and 250,669 (949 Mbp) more than Porechop. Deepbinner's binned reads had a wider range of mean Phred quality scores (q scores) than those from Albacore and Porechop (Table 1). This indicates that Deepbinner's improved recall is largely coming from its ability to bin lower quality reads, though such reads are not 'junk' as they were successfully assigned a ground truth label based on alignment to reference sequences.

Table 1. Classification performance of demultiplexing tools.

	Reads with known ground truth (N=1,642,167)				Other reads		
	Binned reads	Precision (PPV)	Recall (accuracy)	Q score range	Binned unknown	Binned chimeric	Binned neg control
Albacore	81.10%	97.27%	78.89%	6.8–11.3	23.46%	84.27%	**0.00%**
Porechop	85.71%	97.48%	83.55%	6.1–11.3	33.48%	72.42%	1.83%
Deepbinner	**94.84%**	**98.41%**	**93.33%**	4.9–11.3	74.71%	**38.10%**	17.73%
Albacore & Porechop	78.13%	97.79%	76.41%	6.8–11.3	22.51%	63.75%	0.00%
Albacore & Deepbinner	77.66%	98.69%	76.65%	6.8–11.3	22.32%	29.15%	0.00%

Reads with known ground truth = reads which were assigned to a single amplicon reference sequence. Unknown = reads which were unable to be assigned to any amplicon reference. Chimeric = reads which were assigned to more than one amplicon reference. Binned reads = proportion of reads assigned to a barcode. Precision (positive predictive value) = proportion of binned reads correctly assigned. Recall (accuracy) = proportion of all reads correctly assigned. Q score range = mean Phred quality scores of binned reads (2.5th—97.5th percentile). Binned unknown = proportion of unknown reads assigned to a barcode. Binned chimeric = proportion of chimeric reads assigned to a barcode. Binned negative control = proportion of negative control reads assigned to a barcode. When two demultiplexers were combined, reads were only binned if both tools agreed on the classification (otherwise put in the 'none' bin).

Combining multiple demultiplexers (only binning reads where multiple tools agree) improves precision at the cost of recall (Table 1). However, precision never approached 100%, plateauing at about 99% even when we used multiple demultiplexers and considered only high-quality reads.

Deepbinner was much more likely than other demultiplexers to put 'unknown' reads (those which we could not assign to an amplicon, 12.8% of the total reads) into a barcode bin. Some are likely reads that were too low quality to be assigned to a source based on the internal sequence, in which case the classification may be correct. However, the negative control test shows that Deepbinner can suffer from over-sensitivity, putting barcode-less reads in a bin. For chimeric reads (0.31% of the total reads), Deepbinner was less likely than other tools to assign a barcode.

Deepbinner requires as little as 1 CPU and 1 GB of RAM, though it can take advantage of additional resource to run faster, benefitting from up to 16 CPUs and 8 GB of RAM. It is difficult to make a direct comparison between Deepbinner and the other demultiplexing tools, as they use computer resources differently. Albacore is likely the fastest for many users, as its demultiplexing requires little extra time over basecalling. If Deepbinner is run on a CPU, it is the slowest demultiplexer tested ($\sim$15 reads/sec), but when run on a GPU its performance is comparable to Porechop ($\sim$100 reads/sec).

Implications

Many of Deepbinner's advantages stem from the fact that it operates not on basecalled sequences, but on the more informative raw signal. This may account for Deepbinner's higher recall (including the ability to bin lower quality reads) and higher precision. By demultiplexing fast5 files before basecalling, it simplifies downstream analyses such as Nanopolish which require raw reads [9]. The design and implementation of Deepbinner required no knowledge of how a barcode should appear in the raw signal – the neural network learns this from the training data. This opens up the possibility of constructing barcodes with modified DNA bases to increase the size of the genome alphabet. Such high-alphabet barcodes could be easier to differentiate at the signal level, and if so would allow for a greater number of unique barcodes in a given sequence length.

The disadvantages of Deepbinner are similar to those experienced by CNN classifiers in other contexts. Training the network is computationally intensive, and a large volume of training data is required. Trained networks may not generalise well across different flow cells and library preparation kits, necessitating a separate trained network for each. When Deepbinner makes an error during classification (as was often the case in our negative control test), the 'black box' nature of neural networks makes it difficult to understand why.

Recommendations

Deepbinner's high recall is well suited to applications where the most important factor is maximising the number of classified reads. A user can run Deepbinner during sequencing, binning fast5 files as they are produced and run Albacore on each resulting directory of reads. Its greater yield of reads may improve assemblies, even when the additional reads are low-quality [23]. If precision is more important (i.e. minimising the number of incorrectly binned reads), then we recommend using Deepbinner and Albacore in tandem. This can be achieved in the same manner, except with Albacore's demultiplexing enabled during basecalling so users can discard reads where Deepbinner and Albacore disagree.

Our tests showed that demultiplexing precision plateaued at about 99%, i.e. 1% of reads with a ground truth label are consistently assigned to a wrong bin by all demultiplexing tools. This implies that these reads may have the wrong barcode ligated to the DNA, a problem that no demultiplexing tool could fix. This may arise in the library preparation, whereby unligated barcodes could be carried through after sample pooling and then be available for ligation to incorrect DNA fragments in the adapter ligation step. If so, a bead clean-up with size selection after barcode ligation (but before adapter ligation) may mitigate the issue by reducing the number of free barcode sequences. This small amount of barcode crosstalk, along with cross-barcode chimeric reads, is likely inconsequential for isolate sequencing, but it could be a serious problem in some metagenomic or transcriptomic studies [24]. We therefore recommend against using ONT barcoding preparations for quantitative applications until extremely high precision demultiplexing is possible.

Availability and future directions

Deepbinner, documentation and the pre-trained model for native ligation barcoded reads are available on GitHub: https://github.com/rrwick/Deepbinner. The amplicon and negative control datasets used in this manuscript, along with full classification results, are available on figshare: https://figshare.com/projects/Deepbinner/34223.

Future development of Deepbinner will involve training models for other library preparations, such as the ONT SQK-RBK004 Rapid Barcoding Kit. Improved performance and parallelism is also a focus, to ensure that Deepbinner can keep up with high yield sequencing runs in real time.

Deepbinner's false positive rate on barcode-less reads, demonstrated with the negative control read set, may be addressed by modifying the training set. Specifically, including signals which contain adapter sequences but lack a barcode may allow the network to more reliably put such reads in the 'none' bin.

Acknowledgments

We would like to thank Wendy Wilson for conducting the long-range PCR amplification.

References

1. Church GM, Kieffer-Higgins S. Multiplex DNA sequencing. Science. 1988;240(4849):185–188. doi:10.1126/science.3353714.

2. Quick J, Quinlan AR, Loman NJ. A reference bacterial genome dataset generated on the MinION portable single-molecule nanopore sequencer. GigaScience. 2014;3(1):1–6. doi:10.1186/2047-217X-3-22.

3. Jansen HJ, Liem M, Jong-Raadsen SA, Dufour S, Weltzien FA, Swinkels W, et al. Rapid *de novo* assembly of the European eel genome from nanopore sequencing reads. Scientific Reports. 2017;7(1):1–13. doi:10.1038/s41598-017-07650-6.

4. Wick RR, Judd LM, Gorrie CL, Holt KE. Completing bacterial genome assemblies with multiplex MinION sequencing. Microbial Genomics. 2017;3(10):1–7. doi:10.1099/mgen.0.000132.

5. Stoiber M, Brown J. BasecRAWller: Streaming nanopore basecalling directly from raw signal. bioRxiv. 2017; p. 1–15. doi:10.1101/133058.

6. Teng HH, Hall MB, Duarte T, Cao MD, Coin LJM. Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning. bioRxiv. 2017; p. 1–10. doi:10.1101/179531.

7. Boža V, Brejová B, Vinař T. DeepNano: Deep recurrent neural networks for base calling in MinION Nanopore reads. PLOS ONE. 2017;12(6):1–13. doi:10.1371/journal.pone.0178751.

8. Wick R, Judd LM, Holt KE. Comparison of Oxford Nanopore basecalling tools; 2018. Available from: https://doi.org/10.5281/zenodo.1188469.

9. Loman NJ, Quick J, Simpson JT. A complete bacterial genome assembled *de novo* using only nanopore sequencing data. Nature Methods. 2015;12(8):733–735. doi:10.1038/nmeth.3444.

10. Loose M, Malla S, Stout M. Real time selective sequencing using nanopore technology. bioRxiv. 2016;13(9):038760. doi:10.1101/038760.

11. Krizhevsky A, Sutskever I, Hinton GE. ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems. 2012; p. 1097–1105. doi:http://dx.doi.org/10.1016/j.protcy.2014.09.007.

12. Szegedy C, Ioffe S, Vanhoucke V, Alemi A. Inception-v4, Inception-ResNet and the impact of residual connections on learning. CoRR. 2016;abs/1602.0. doi:10.1016/j.patrec.2014.01.008.

13. Zeiler MD, Fergus R. Visualizing and understanding convolutional networks. In: Fleet D, Pajdla T, Schiele B, Tuytelaars T, editors. Computer Vision – ECCV 2014. Cham: Springer International Publishing; 2014. p. 818–833.

14. Jia Deng, Wei Dong, Socher R, Li-Jia Li, Kai Li, Li Fei-Fei. ImageNet: A large-scale hierarchical image database. 2009 IEEE Conference on Computer Vision and Pattern Recognition. 2009; p. 248–255. doi:10.1109/CVPRW.2009.5206848.

15. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. TensorFlow: A system for large-scale machine learning. 12th USENIX Symposium on Operating Systems Design and Implementation. 2016; p. 265–284. doi:10.1038/nn.3331.

16. Chollet F. Keras; 2015. Available from: https://github.com/keras-team/keras.

17. Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition. International Conference on Learning Representations (ICRL). 2015; p. 1–14. doi:10.1016/j.infsof.2008.09.005.

18. Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, et al. Going deeper with convolutions. Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition. 2015;07-12-June:1–9. doi:10.1109/CVPR.2015.7298594.

19. Ioffe S, Szegedy C. Batch normalization: accelerating deep network training by reducing internal covariate shift. CoRR. 2015;abs/1502.0. doi:10.1007/s13398-014-0173-7.2.

20. Lin M, Chen Q, Yan S. Network in network. CoRR. 2013;abs/1312.4:1–10. doi:10.1109/ASRU.2015.7404828.

21. Perez L, Wang J. The effectiveness of data augmentation in image classification using deep learning. CoRR. 2017;abs/1712.0.

22. Li H. Minimap2: Pairwise alignment for nucleotide sequences. Bioinformatics. 2018; p. bty191. doi:10.1093/bioinformatics/bty191.

23. Wick RR, Judd LM, Gorrie CL, Holt KE. Unicycler: Resolving bacterial genome assemblies from short and long sequencing reads. PLOS Computational Biology. 2017;13(6):e1005595. doi:https://doi.org/10.1371/journal.pcbi.1005595.

24. Sinha R, Stanley G, Gulati GS, Ezran C, Travaglini KJ, Wei E, et al. Index switching causes ”spreading-of-signal” among multiplexed samples in Illumina HiSeq 4000 DNA sequencing. bioRxiv. 2017; p. 125724. doi:10.1101/125724.

Supporting information

File S1. Confusion matrices. Classification counts per reference bin and predicted bin, for each demultiplexer tested. Includes per-bin values for precision and recall.

File S2. Amplicon sequences. FASTA file of the 12 amplicon sequences.

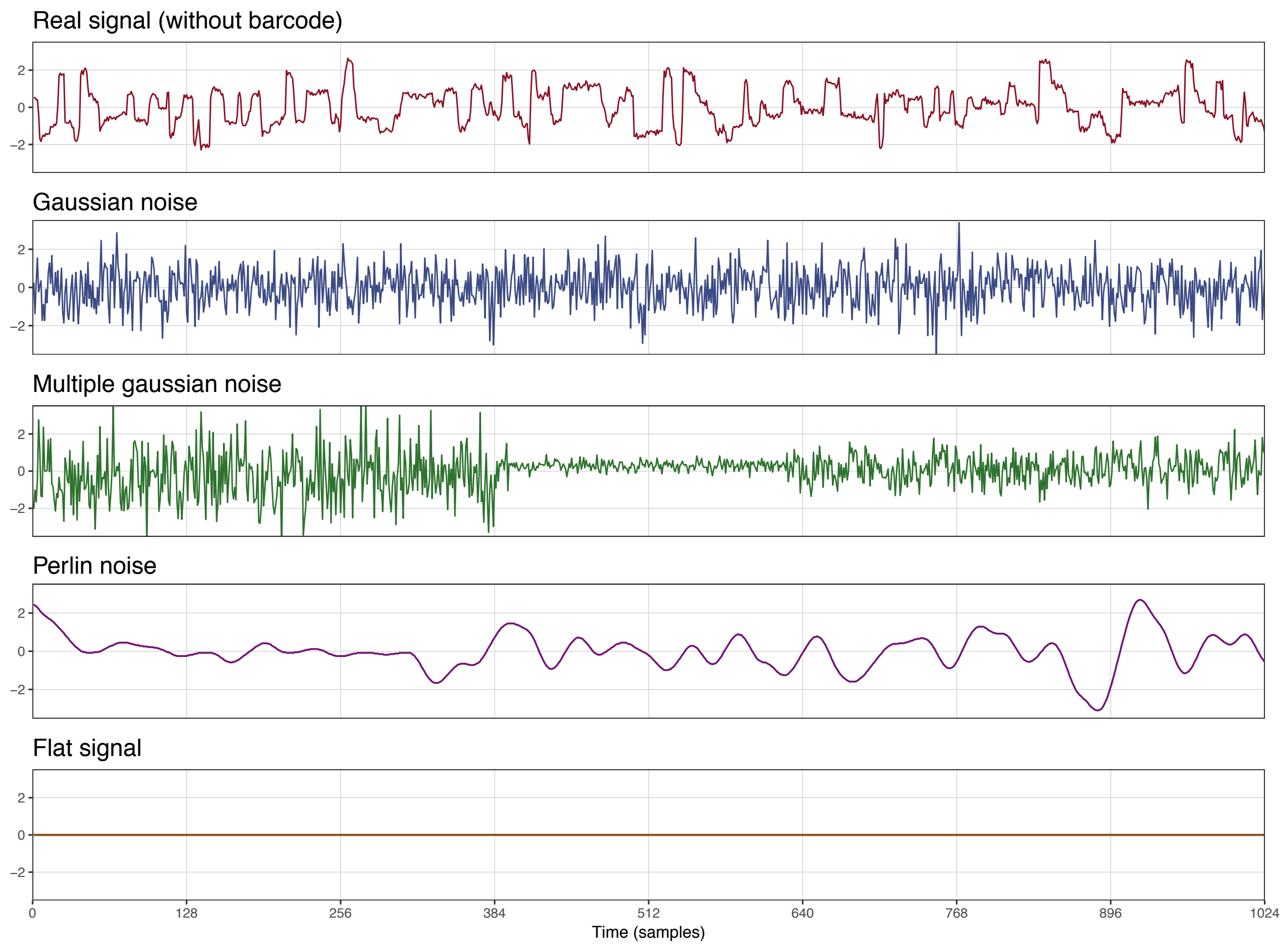

Figure S1. No-barcode training signals. Multiple types of signals were included in the training set to explicitly teach the neural network what a barcode-free signal looks like. The signal amplitude has been normalised to a mean of 0 and a variance of 1.

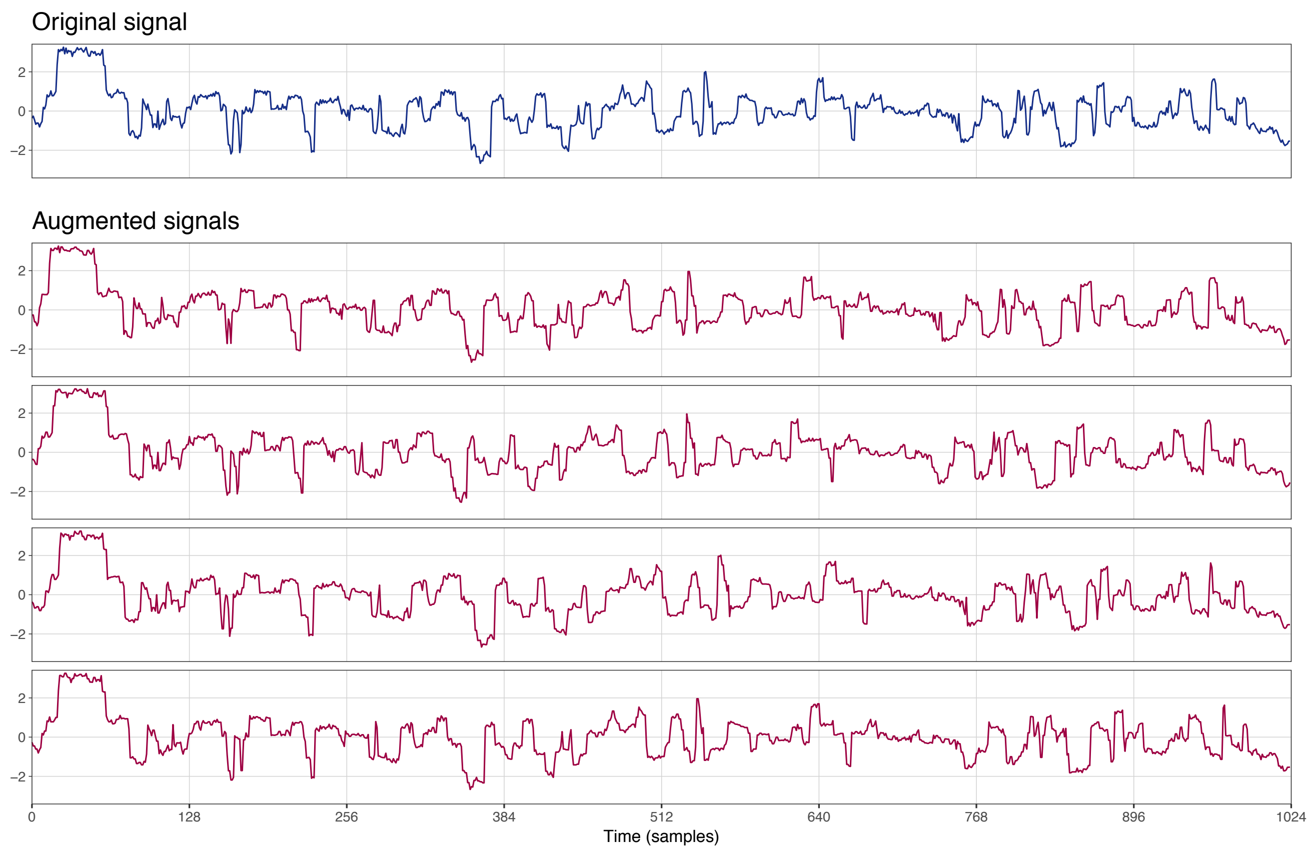

Figure S2. Data augmentation via temporal distortion. One real training sample can yield multiple additional training samples by distorting the signal along the temporal axis. The signal amplitude has been normalised to a mean of 0 and a variance of 1.

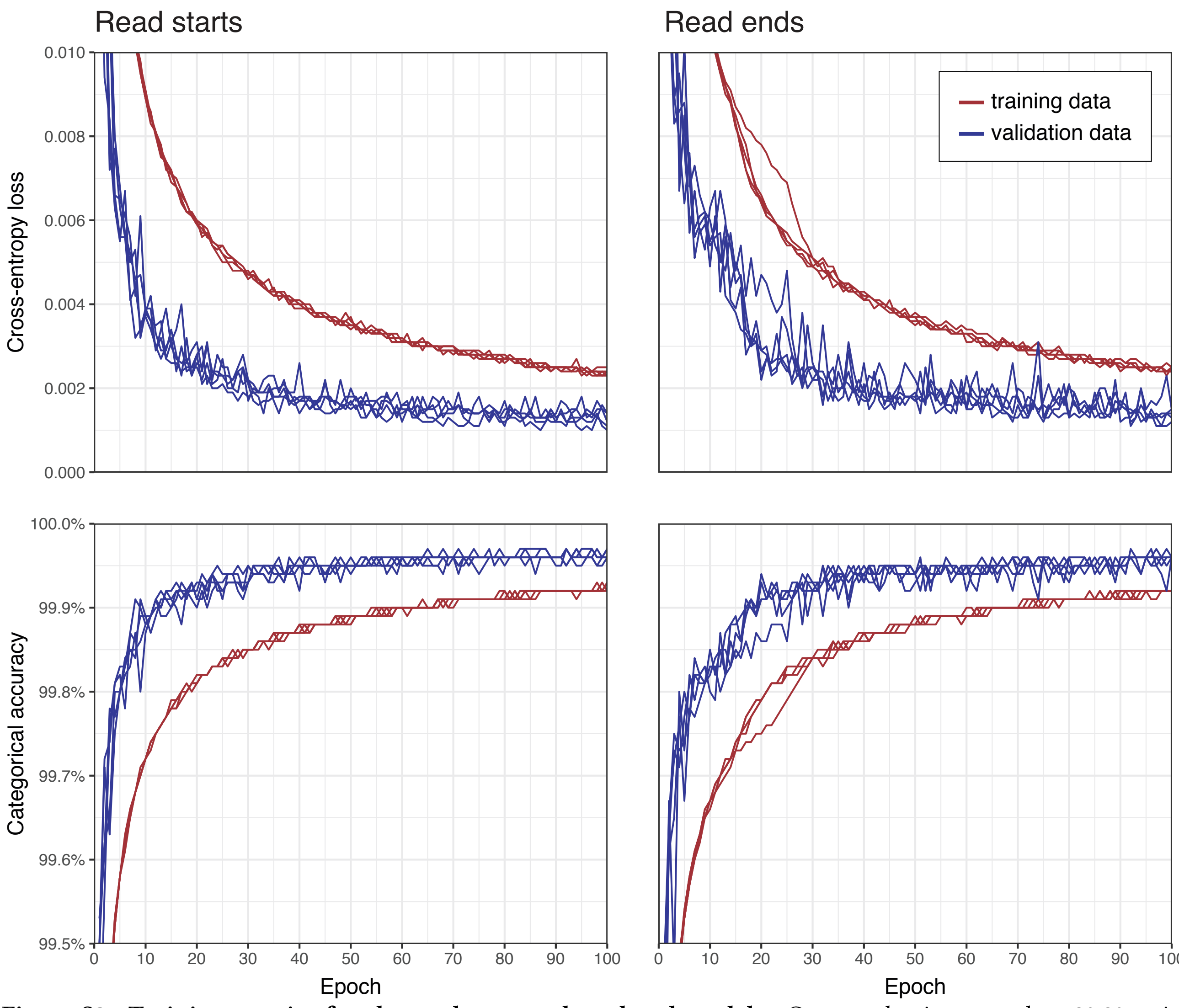

Figure S3. Training metrics for the read start and read end models. Generated using a random 80:20 training:validation split in five replicates. Note that training data has poorer performance than validation data due to data augmentation and training-only layers in the network (Gaussian noise and dropout layers).