Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning

Haotian Teng[1,*], **Michael B. Hall**[1], **Tania Duarte**[1], **Minh Duc Cao**[1], **and Lachlan J.M. Coin**[1,*]

[1]Institute for Molecular Bioscience, University of Queensland, St Lucia, Brisbane, QLD 4072 Australia
[*]Correspondence: haotian.teng@uq.net.au, l.coin@imb.uq.edu.au

ABSTRACT

Sequencing by translocating DNA fragments through an array of nanopores is a rapidly maturing technology which offers faster and cheaper sequencing than other approaches. However, accurately deciphering the DNA sequence from the noisy and complex electrical signal is challenging. Here, we report the first deep learning model - Chiron - that can directly translate the raw signal to DNA sequence, without the error-prone segmentation step. We show that our model provides state-of-the-art basecalling accuracy when trained with only a small set of 4000 reads. Chiron achieves basecalling speeds of over 2000 bases per second using desktop computer graphics processing units, making it competitive with other deep-learning-based basecalling algorithms.

Introduction

DNA sequencing via bioengineered nanopores, recently introduced to the market by Oxford Nanopore Technologies (ONT), has profoundly changed the landscape of genomics. A key innovation of the ONT nanopore sequencing device, MinION, is that it measures the changes in electrical current across the pore as a single-stranded molecule of DNA passes through it. The device then uses the signal to determine the nucleotide sequence of the DNA strand[1–3]. Importantly, this signal can be obtained and analysed by the user while the sequencing is still in progress. Because of the minuscule size of these pores, a large number can be packed into an array, meaning the MinION device is the size of a stapler, making it extremely portable. The small size and real-time nature of the sequencing opens up new opportunities in time-critical genomics applications[4–6] and in remote regions[7–11].

While nanopore sequencing can be massively scaled up by designing large arrays of nanopores and allowing faster translocation of DNA fragments, one of the bottle-necks in the pipeline is basecalling, which translates the electrical raw signals to a nucleotide sequence.

DeepNano introduced the idea of basecalling using a bi-directional Recurrent Neural Network (RNN)[12]. DeepNano first uses a segmentation algorithm to detect events in the raw signal, from which the mean, standard deviation and length are provided as input to the DeepNano RNN. ONT have also developed a RNN basecaller - nanonet - which also relies on an initial event segmentation. The nanonet RNN predicts the probability of each k-mer for the segment. As k-mers from successive segments are expected to overlap by k-1 bases, a dynamic programming algorithm is applied to find the most probable path, which results in the basecalled sequence data. Additionally, ONT provide Albacore, their proprietary basecaller. Albacore is considered the 'gold standard' in terms of accuracy, but as it is not open source, we cannot comment on it's implementation.

BasecRAWller[13] defers the segmentation step until after an initial unidirectional RNN has been run to analyse sequence content. After an intermediate segmentation step, a final unidirectional RNN is used to infer the base-called sequence data. By utilising a pair of unidirectional RNNs, basecRAWller is able to process the raw signal data in a stream.

In this article we present a new deep neural network model. Chiron has a novel architecture which couples a convolutional neural network (CNN) with an RNN and a Connectionist Temporal Classification (CTC) decoder[14]. This enables it to model the raw signal data directly, without use of an event segmentation step. It is the first neural network which can translate raw electrical signal directly to nucleotide sequence. Chiron is trained on only a small subset of data from a simple virus genome and an *E. coli* sample and yet it is able to generalise to larger genomes such as other bacteria and human. Chiron is as accurate as the ONT designed and trained Albacore in some settings, and outperforms all other existing methods.

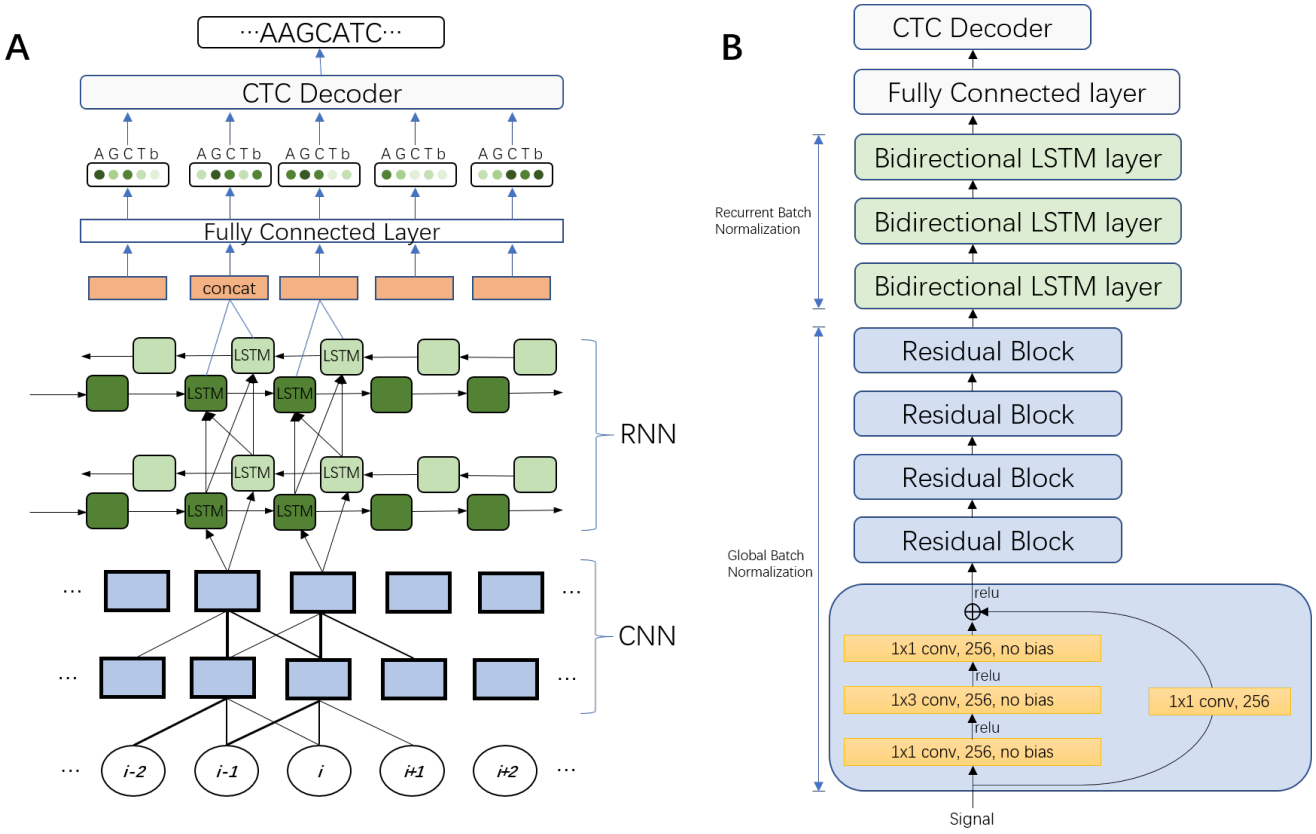

Figure 1. A) An unrolled sketch of the neural network architecture. The circles at the bottom represent the time series of raw signal input data. Local pattern information is then discriminated from this input by a CNN. The output of the CNN is then fed into a RNN to discern the long-range interaction information. A fully connected layer is used to get the base probability from the output of the RNN. These probabilities are then used by a CTC decoder to create the nucleotide sequence. The repeated component is omitted. **B)** Final architecture of the Chiron model. We explored variants of this architecture by varying the number of convolutional layers from 3 to 10 and recurrent layers from 3 to 5. We also explored networks with only convolutional layers or recurrent layers, **1×3 conv, 256, no bias** means a convolution operation with a 1×3 filter and a 256 channels output with no bias added. Further definitions can be found in Methods.

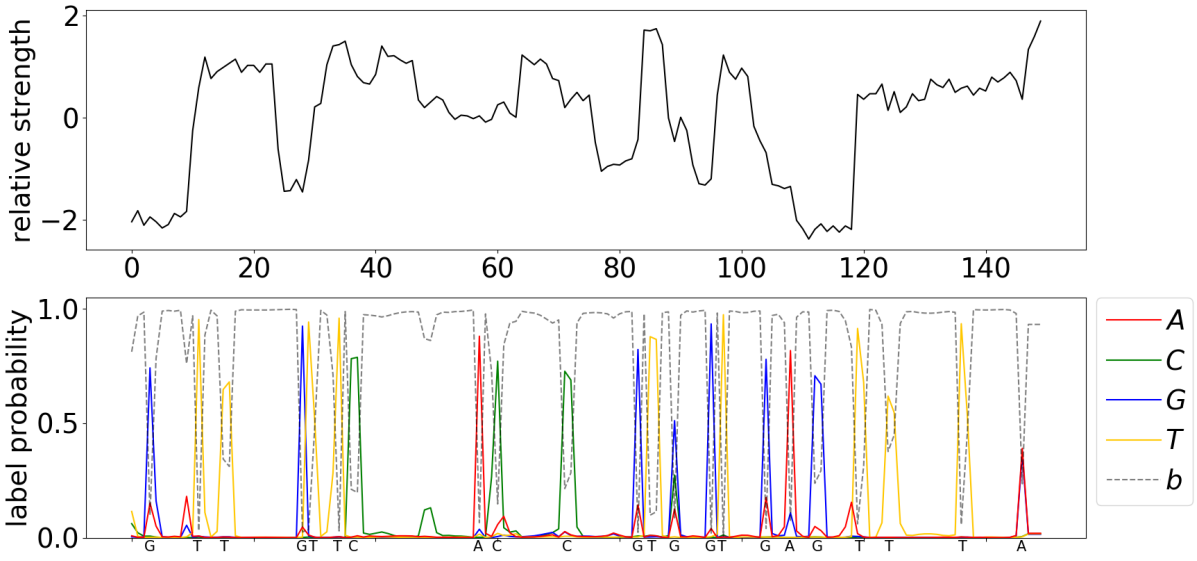

Figure 2. Visualization of the predicted probability of bases and the readout sequence. The upper pane is a normalised raw signal from the Minion Nanopore sequencer, normalised by subtracting the mean of the whole signal and then dividing by the standard deviation. The bottom pane shows the predicted probability of each base at each position from Chiron. The final output DNA sequence is annotated on the x-axis of the bottom plane.

Results

Deep neural network architecture

We have developed a deep neural network (NN) for end-to-end, segmentation-free basecalling (shown in Figure 1) which consists of two sets of layers: a set of convolutional layers and a set of recurrent layers. The convolutional layers discriminate local patterns in the raw input signal, whereas the recurrent layers integrate these patterns into basecall probabilities. At the top of the neural network is a CTC decoder[14] to provide the final DNA sequence according to the base probabilities. More details pertaining to the NN are provided in Methods.

The NN we present is an end-to-end basecaller, in the sense that it predicts a complete DNA sequence from a raw signal. However, this would become computationally infeasible for full-length nanopore reads. To make this model more computationally efficient, we run it on overlapping windows consisting of 300 raw signals, roughly corresponding to around 10-20bp (which we call slices), which are processed in parallel. The output of these sliding windows are overlapped, and a consensus call is used to generate the final reported base at each position.

Performance Comparison

CHIRON was trained on a subset of data from two samples: the Phage Lambda from ONT (*Escherichia virus Lambda* - NCBI Reference Sequence: NC_001416.1) and an *Escherichia coli* (K12 MG1655) sample. The performance of the model was assessed by the identity rate (see Table 1) on a set of testing datasets. All samples used for training and testing were sequenced using the same 1D protocol on R9.4 flowcells (See Methods). For evaluation, we used the remaining reads from the samples used in testing (Phage Lambda and *E. coli*), plus a *Mycobacterium tuberculosis* sample, and a small subset of human data from chromosome 21 part 3 from the Nanopore WGS Consortium[15] (see Table 3).

In order to establish the ground-truth of the data, we sequenced and assembled the *E. coli* and *M. tuberculosis* samples using Illumina technology (see Methods) which provided high per base accuracy assembly. The reference sequence for the Phage Lambda virus is NCBI Reference Sequence NC_001416.1 and for the human data the GRCh38 reference was used.

While we obtained 34,383 reads for Phage Lambda and 15,012 reads for *E. coli* from the MinION sequencing runs (see Table 3), we used only 2,000 reads from each for training. We labelled the raw signals by identifying the raw signal segment corresponding to the nucleotide assumed to be in the pore at a given time-point (see Methods). It took the model 10 hours to train 3 epoch with 4,000 reads ($\sim$ 4Mbp) on a Nvidia K80 GPU.

Table 1 presents the accuracy of the four basecalling methods on the data. Chiron has the highest identity rate on the *E. coli* sample, and the equal highest on *M. tuberculosis*. In the cases where Chiron did not have the highest identity rate (Lambda

Dataset	Basecaller	Deletion Rate	Insertion Rate	Mismatch Rate	**Identity Rate**
Lambda	Metrichor	0.0893	0.0238	0.0457	0.8650 (-0.0246)
	Albacore	**0.0635**	0.0382	0.0469	**0.8896**
	BasecRAWller	0.0789	0.1001	0.1056	0.8154 (-0.0742)
	Chiron	0.0820	**0.0213**	**0.0403**	0.8776 (-0.012)
E. coli	Metrichor	0.0752	0.0193	0.0384	0.8864 (-0.0193)
	Albacore	**0.0576**	0.0327	0.0414	0.901 (-0.0047)
	BasecRAWller	0.0716	0.1040	0.1030	0.8254 (-0.0803)
	Chiron	0.0636	**0.0181**	**0.0307**	**0.9057**
M. tuberculosis	Metrichor	0.0763	**0.0240**	0.0435	0.8802 (-0.0117)
	Albacore	**0.0612**	0.0357	0.0468	**0.8919**
	BasecRAWller	0.0717	0.1085	0.1042	0.8241 (-0.0678)
	Chiron	0.0716	0.025	**0.0433**	0.8851 (-0.0068)
Human	Metrichor	0.1295	**0.0415**	0.0765	0.794 (-0.0446)
	Albacore	0.0862	0.0651	**0.0752**	**0.8386**
	BasecRAWller	**0.0841**	0.1028	0.101	0.8149 (-0.0237)
	Chiron	0.0913	0.0514	0.0933	0.8154 (-0.0232)

Table 1. Results from the experimental validation and benchmarking of Chiron against three other Nanopore basecallers. *Identity rate* is defined as the number of matched bases divided by the number of bases in the reference genome for that sample (the higher the better), while *Deletion/Insertion/Mismatch rate* are defined as the number of deleted/inserted/mismatched bases divided by the number of bases in the reference genome (the lower the better). This statistic effectively summarises the basecalling accuracy of the associated model.

	CPU rate (GPU rate) bp/s
Albacore	2975
BasecRAWller	81
Chiron	21 (1652)

Table 2. Speed benchmarking for basecalling with Chiron and two other local basecallers. *Rate* is calculated by dividing the number of nucleotides basecalled by the total CPU time for the basecalling analysis. Chiron is also capable of running on a GPU and its rate in this mode is included in parentheses. The reported rate is the average for that basecaller across all samples analysed. A Nvidia K80 GPU is used for the GPU testing of Chiron. Albacore is not capable of running in GPU mode, while BasecRAWller was no faster when running with GPU (data not shown).

and Human) is was no more than 0.0232 from the best. Additionally, it had the lowest mismatch rate on Lambda and *E. coli* (0.0403 and 0.0307).

In terms of speed on a CPU processor, Chiron is slower (21bp/s) than Albacore (2975bp/s) and - to a lesser extent - BasecRAWller (81bp/s). However, when run on a Nvidia K80 GPU, we achieved a basecalling rate of 1652bp/s. (We also tested Chiron on a Nvidia GTX 1080 Ti GPU and obtained a rate of 2657bp/s). As Albacore does not have the ability to be run on GPU and basecRAWller, in our hands, gained no speed improvement with GPU we could not compare them to Chiron in this mode. Metrichor was not included in the speed benchmarking as it is not possible to gather information about CPU/GPU speed. We also found that Chiron was faster than Nanonet on GPU (data not shown).

Discussion

Segmenting the raw nanopore electrical signal into piece-wise constant regions corresponding to the presence of different k-mers in the pore is an appealing but error-prone approach. Segmentation algorithms determine a boundary between two segments based on a sharp change of signal values within a window. The window size is determined by the expected speed of the translocation of the DNA fragment in the pore. We noticed that the speed of DNA translocation is variable during a sequencing run, which coupled with the high level of signal-to-noise in the raw data, can result in low segmentation accuracy. As a result, the segmentation algorithm often makes conservative estimates of the window size, resulting in segments smaller than the actual signal group for k-mers While dynamic programming can correct this by joining several segments together for a k-mer, this effects the prediction model.

All existing nanopore base callers prior to Chiron employ a segmentation step. The first nanopore basecalling algorithms[16,17] employed a Hidden Markov Model, which maintains a table of event models for all possible k-mers. These event models were learned from a large set training data. More recent methods (DeepNano[12], nanonet) train a deep neural network for inferring k-mers from segmented raw signal data.

A recent basecaller named BasecRAWller[13] used an initial neural network (called a *raw* network) to output probabilities of boundaries between segments. A segmentation algorithm is then applied to segment these probabilities into discrete events. BasecRAWller then uses a second neural network (called the *fine-tune* network) to translate the segmented data into the base sequence.

Our proposed model is a departure from the above approaches in that it performs base prediction directly from raw data without segmentation. More-over the core-model is an end-to-end basecaller in the sense that it predicts the complete base sequence from raw signal. This is made possible by combining a multi-layer convolutional neural network to extract the local features of the signal, with a recurrent neural network to predict the probability of nucleotides in the current position. Finally, the complete sequence is called by a simple greedy algorithm, based on a typical CTC-style decoder[14], reading out the nucleotide in each position with the highest probability. Thus, the model need not make any assumption of the speed of DNA fragment translocation and can avoid the errors introduced during segmentation.

To improve the basecalling speed and to minimize its memory requirements, we run the neural network on a 300-signal sliding window (equivalent to approximately 30bp), overlapping the sequences on these windows and generating a consensus sequence. Chiron has the potential to stream these input raw signal 'slices' into output sequence data, which will become increasingly important aspect of basecalling very long reads (100kb+), particularly if used in conjunction with the read-until capabilities of the MinION.

Our model was either the best or second-best in terms of accuracy on all of the datasets we tested. This includes the human dataset, despite the fact that the model has not seen human DNA during training. Our model has only been trained on a mixture of 2,000 bacterial and 2,000 viral reads. The most accurate basecaller is the proprietary ONT Albacore basecaller. Chiron is within 1% accuracy on bacterial DNA, but only within 2% accuracy on human DNA. More extensive training on a broader spectrum of species, including human can be expected to improve the performance of our model. There are also improvements in accuracy to be gained from a better alignment of overlapping reads and consensus calling. Increasing the size of the sliding window will also improve accuracy but at the cost of increased memory and running time.

Our model is substantially more computationally expensive than Albacore and somewhat more computationally expensive than BasecRAWller when run in CPU mode. This is to be expected given the extra depth in the neural network. Nevertheless, our model can be run in a GPU mode, which makes it competitive with Albacore (which does not have a GPU mode), and faster than Nanonet in GPU mode. Our method can be further sped up by increasing the sliding window step size, although this may impact accuracy.

Conclusion

We have presented a novel deep neural network approach for streaming segmentation free basecalling of raw Nanopore signal. Our approach is the only method that can map the raw signal data directly to base sequence without segmentation. We trained our method on 4000 reads sequenced from the simple genome lambda virus and E.coli, but the method is sufficiently generalised to be able to base call data from other species. Our method has state-of-art accuracy - outperforming the ONT cloud basecaller Metrichor as well as another 3rd-party basecaller, BasecRAWller. With GPU acceleration, our method is also faster than current data collection speed, so that it can support real-time basecalling.

Methods

Deep neural network architecture

Our model combines a 5-layer CNN[18] with a 3-layer RNN and a fully connected network (FNN) in the last layer that calculates the probability for a CTC decoder to get the final output. This structure is similar to that used in speech recognition[19]. We found that both the CNN and RNN layers are essential to the base calling as removing either will cause a dramatic drop in prediction accuracy, which is described more in the Training section.

Preliminaries Let a raw signal input with T time-points $\mathbf{s} = [s_1, s_2, ..., s_T]$ and the corresponding DNA sequence label (with K bases) $\mathbf{y} = [y_1, y_2, ..., y_K]$ with $y_i \in \{A, G, C, T\}$ be sampled from a training dataset $\chi = \{(\mathbf{s}^{(1)}, \mathbf{y}^{(1)}), (\mathbf{s}^{(2)}, \mathbf{y}^{(2)}), ...\}$. Our network directly translates the input signal time series s to the sequence y without any segmentation steps.

We normalise the input signal by subtracting the mean of the whole read and dividing by the standard deviation. $\mathbf{s'} = (\mathbf{s} - \bar{s})/std(s)$.

Then we feed the normalised signal into a residual block[20] combined with batch normalisation[21] in the 5 convolution layers to extract the local pattern from the signal. We set the stride as 1 to ensure the output of the CNN has the same length of the input raw signal. The residual block is illustrated in Figure 1, a convolution operation with a l×m filter, n×p stride and s output channels on a k channels input is defined as:

$$Output(i, j, s) = \sum_{di<l, dj<m, q<k} Input(i \cdot n + di, j \cdot p + dj, q) \cdot Filter(di, dj, q, s)$$

.

An activation operation is performed after the convolution operation. Various kinds of activation functions can be chosen, however, in this model we use a Rectified Linear Unit (ReLU) function in the activation operation which has been reported to have a good performance in CNN, defined as :

$$ReLU(x) = \max(x, 0)$$

Following the convolution layers are multiple bi-directional RNN layers[22]. We use a LSTM cell[23] with a separate batch normalisation on the inside cell state and input term[24].

A typical batch normalisation procedure[21] is

$$BN(\mathbf{x};\gamma,\beta) = \beta + \gamma \odot \frac{\mathbf{x} - \hat{E}[\mathbf{x}]}{\sqrt{\hat{Var}[\mathbf{x}] + \varepsilon}}, \tag{1}$$

where $\mathbf{x}$ be a inactivation term.

Let h_t^l be the output of l_{th} RNN layer at time t, the batch normalisation for a LSTM cell is

$$(\mathbf{f}_t, \mathbf{i}_t, \mathbf{o}_t, \mathbf{g}_t) = BN(\mathbf{W}_h \mathbf{h}_{t-1}^l; \gamma_h, \beta_h) + BN(\mathbf{W}_x \mathbf{h}_t^{l-1}; \gamma_x, \beta_x) + \mathbf{b} \tag{2}$$

$$\mathbf{c}_t = \sigma(\mathbf{f}_t) \odot \mathbf{c}_{t-1} + \sigma(\mathbf{i}_t) \odot tanh(\mathbf{g}_t) \tag{3}$$

$$\mathbf{h}_t = \sigma(\mathbf{o}_t) \odot tanh(BN(\mathbf{c}_t; \gamma_c, \beta_c)) \tag{4}$$

The batch normalisation is calculated separately in the recurrent term $\mathbf{W}_h \mathbf{h}_{t-1}^l$ as well as the input term $\mathbf{W}_x \mathbf{h}_t^{l-1}$. The parameters β_h and β_x are set to zero to avoid the redundancy with $\mathbf{b}$. The last forward layer $\vec{h}_{if}^L$ and the backward layer $\vec{h}_{ib}^L$ are concatenated together as an input to a fully connected layer

$$\mathbf{H}_i = [\mathbf{h}_{iw}^L, \mathbf{h}_{ib}^L]. \tag{5}$$

The final output is transferred through a fully connected network followed by a softmax operation

$$p(\mathbf{o}_i = j) = \frac{\exp \mathbf{W}_j \mathbf{H}_i}{\sum_j \exp \mathbf{W}_j \mathbf{H}_i} \tag{6}$$

The output $\mathbf{o}_i, i = 1, 2, ..., T$ predict the symbol given the input vector $\mathbf{x}$, $P(o_i = j|\mathbf{x})$. If the read is a DNA sequence then $j \in \{A, G, C, T, b\}$, where b represents a blank symbol(Figure 1). We then calculate the CTC loss between the output sequence $\mathbf{o}$ with label $\mathbf{y}$[14].

Convolutional network to extract local patterns: We use 256 channel filters for all five convolutional layers. In each layer we use a residual block[20] (Figure 1) composed of two branches. A 1x1 filter is used for reshaping in the first branch. In the second branch, a 1x1 convolution filter is followed by a rectified linear unit (RELU)[25] activation function and a 1x3 filter with a RELU activation function as well as a 1x1 filter. All filters have the same channel number of 256. An element-wise addition is performed on the two branches followed by a RELU activation function. A global batch normalisation operation is added after every convolution operation. We tried a large kernel size (5,7,11) and different channel numbers (128,1024), and found the above combination yielded the best performance.

Recurrent layers for unsegmented labelling: The local pattern extracted from the CNN described above is then fed to a 3-layer RNN (Figure 1). Under the current ONT sequencing settings, the DNA fragments translocate through the pore with a speed of roughly 250 or 450 bases per second, depending on the sequencing chemistry used, while the sampling rate is 4000 samples per second. Because the sampling rate is higher than the translocation rate, each nucleotide usually stays in the current position for about 5 to 15 samplings, on average. Furthermore, as a number of nearby nucleotides also influence the current, 40 to 100 samples (based on a 4- or 5-mer assumption) could contain information about a particular nucleotide. We used a 3-layer bidirectional recurrent neural network for extracting this long range information. LSTM (Long Short Term Memory) cells[26,27] with 200 hidden units are used in every layer and a fully connected neural network (FNN) is used to translate the output from the last RNN layer into a prediction. The output of the FNN is then fed into a CTC decoder to obtain the predicted nucleotide sequence for the given raw signals. An Adam optimizer[28] with an initial learning rate of 0.001 is used to minimize the CTC loss.

Sample	No. reads	Median read length (bp)
Phage Lambda	34,383	5720
E. coli	15,012	5,836
M. tuberculosis	147,594	3,423
Human	10,000	6,154

Table 3. Details about the number of reads and their median read length for data that was used in evaluation of the various basecallers.

Improving basecalling performance: To achieve a better accuracy and less memory allocation, we apply a sliding window (default of 300 raw signals), with a pre-set sliding step size (default of 10% of window size), to the long raw signal. This gives a group of short reads with uniform length (window length) that overlap the original long read. We then apply basecalling in parallel on these short reads, and reassemble the whole DNA sequence by finding the maximum overlap between two adjacent short reads, and read out the consensus sequence. Note here the reassembly is very easy because the order of the short reads is known. This procedure improves the accuracy of the basecalling and also enables parallel processing on one read.

Data preparation

Sequencing: The library preparations of the *E. coli* and *M. tuberculosis* samples were done using the *1D gDNA selecting for long reads using SQK-LSK108* (March 2017 version) protocol with the following modifications. Increase the incubation time to 20 minutes in each end-repair and ligation step; use 0.7x Agencourt^R AMPure^R XP beads (Beckman Coulter) immediately after the end-repair step and incubation of the eluted beads for 10 minutes; and use elution buffer (ELB) warmed up at 50^{o}C with the incubation of the eluted bead at the same temperature. For the Lambda sample, the *1D Lambda Control Experiment for MinION^{TM} device using SQK-LSK108* (January 2017 version) protocol was followed with some changes: sheared the sample at 4000rpm (2x1 minutes); 30 minutes of incubation in each end-repair step and 20 minutes for adaptor ligation and elution of the library with 17μL of ELB. All samples were sequenced on new FLO-MIN106, version R9.4, flow cells with over 1100 active single pores and the phage was sequenced in a MinION Mk1 (232ng in 6h run) while the bacteria samples were sequenced in a MinION Mk1B (1μg *E. coli* and 595ng *M. tuberculosis* in 22h and 44h runs, respectively). The *E. coli* sample was run on the MinKNOW version 1.4.3 and the other samples in earlier versions of the software.The *E. coli* sample was also sequenced on Illumina MiSeq using paired-end 300x2 to 100-fold coverage. An assembly of the *E. coli* genome was constructed by running Spades[29] on the MiSeq sequencing data of the sample. The genome sequence of the Phage Lambda is NCBI Reference Sequence: NC_001416.1.

Labelling of raw signal: We used Metrichor, the basecaller provided by ONT which runs as a cloud service, to basecall the MinION sequencing data. We then utilised a modified version of nanoraw[30] for labelling of the data. Briefly, we aligned the basecalled sequence data to the genome of the sample. From the alignment, we could correct the errors introduced by Metrichor, and map the corrected data back to the raw data. The resulting labelling consists of the raw signal data, as well as the boundaries of raw signals when the DNA fragment translocates to a new base.

Training dataset We created a training set using 2,000 reads from *E. coli* and 2,000 reads from Phage Lambda. In every start of the training epoch, the dataset is shuffled. Training on this mixture dataset gave the model better performance both on generality and accuracy (see Table 1).

Training

We use the labelling from Metrichor to train, although our neural network architecture is translation invariant and not restricted by the sequence length, a uniform length of sequences is suited for batch feeding, thus can accelerate the training process. We cut the original reads into short segments with a uniform length of 200, 400 and 1000, and trained on these batches in alternation. We tested several different architectures of the neural network, (see Table 4) with the CNN-RNN network architecture having the best accuracy compared to a CNN- or RNN-only network. Also using more layers seems to increase the performance of the model, however, the time consumed for training and basecalling is also increased. In the final structure we use 5 convolution layers and 3 recurrent layers, as adding layers above this structure gave negligible performance improvement.

Parameters for basecalling

All basecallers were invoked on the same set of reads for each sample. When using our model to basecall, we first slice the raw signal with a 300 length window, and sliding the window by 30, and then feed the base caller a batch of the 300 length segment signal with a batch size equal to 1100, and then we simply assemble the short reads by a pair-wise alignment between neighbouring reads, and output the consensus sequence from the alignment. All basecalling with Albacore (version

Architecture	normalised edit distance
3 Convolutional Layers	0.4007 ± 0.0277
5 Convolutional Layers	0.3903 ± 0.0230
10 Convolutional Layers	0.3874 ± 0.0186
3 Bidirectional Recurrent Layers	0.2987 ± 0.0221
5 Bidirectional Recurrent Layers	0.2930 ± 0.0215
3 Convolutional Layers + 3 Bidirectional Recurrent Layers	0.2011 ± 0.0252
5 Convolutional Layers + 5 Bidirectional Recurrent Layers	0.2001 ± 0.0177

Table 4. Comparison of normalised edit distance with different neural network architectures. The normalised edit distance is the edit distance between predicted reads and labelled reads and normalised by segments length.

1.1.1) and BasecRAWller[13] (version 0.1) was done with default parameters. For the configuration setting in Albacore, `r94_450bps_linear.cfg` was used for all samples, as this matches the flowcell and kit used for each sample.

Comparison

To assess the performance of each program, the resulting FASTA/FASTQ file from basecalling was aligned to the reference genome using `graphmap`[31] with the default parameters. The resulting BAM file is then assessed by the `japsa` error analysis tool (`jsa.hts.errorAnalysis`) which looks at the deletion, insertion, and mismatch rates, the number of unaligned and aligned reads, and the identification rate compared to the reference genome. The identity rate is calculated as $\frac{number\ of\ matched\ bases}{number\ of\ bases\ in\ reference}$ and is the marker used here for basecalling accuracy.

Data availability

Sequencing data in this study are in the process of being deposited to the European Nucleotide Archive (ENA). Program and code are available at https://github.com/haotianteng/chiron pypi package index 0.1.2 at https://pypi.python.org/pypi/chiron.

Authors' contributions

MH, MDC and LC conceived the study and designed the experimental framework. HTT designed and implemented the Chiron algorithm. MDC, LC and TD designed and performed the MinION sequencing. MDC labelled the training data. HTT and MH ran the performance comparison. HTT and MDC wrote the initial draft. HTT, MH and LC refined the manuscript. All authors contributed to editing the final manuscript.

Competing financial interests

LC is a participant of Oxford Nanopore's MinION Access Programme (MAP) and received the MinION device, MinION Flow Cells and Oxford Nanopore Sequencing Kits in return for an early access fee deposit. LC and MDC received travel and accommodation expenses to speak at an Oxford Nanopore-organised conference. None of the authors have any commercial or financial interest in Oxford Nanopore Technologies Ltd.

Acknowledgements

LC was supported by an ARC Future Fellowship (FT110100972). The research is supported by funding from the Institute for Molecular Bioscience Centre for Superbugs Solutions (610246). MH is supported by a Westpac Future Leaders Scholarship (2016) awarded by the Westpac Bicentennial Foundation. We thank Jianhua Guo for contributing the DNA for the *E. coli* sample. We thank Arnold Bainomugisa for extracting DNA for the *M. tuberculosis* sample. We thank Sheng Wang and Han Qiao for the helpful discussion.

References

1. Kasianowicz, J. J., Brandin, E., Branton, D. & Deamer, D. W. Characterization of individual polynucleotide molecules using a membrane channel. *Proceedings of the National Academy of Sciences* **93**, 13770–13773 (1996). URL http://www.pnas.org/content/93/24/13770.full. DOI 10.1073/pnas.93.24.13770.

2. Branton, D. *et al.* The potential and challenges of nanopore sequencing. *Nature biotechnology* **26**, 1146–53 (2008). URL http://dx.doi.org/10.1038/nbt.1495. DOI 10.1038/nbt.1495.

3. Stoddart, D., Heron, A. J., Mikhailova, E., Maglia, G. & Bayley, H. Single-nucleotide discrimination in immobilized DNA oligonucleotides with a biological nanopore. *Proceedings of the National Academy of Sciences of the United States of America* **106**, 7702–7 (2009). URL http://www.pnas.org/content/106/19/7702.abstract. DOI 10.1073/pnas.0901054106.

4. Ashton, P. M. *et al.* MinION nanopore sequencing identifies the position and structure of a bacterial antibiotic resistance island. *Nature Biotechnology* **33**, 296–300 (2014). URL http://www.nature.com/doifinder/10.1038/nbt.3103. DOI 10.1038/nbt.3103.

5. Cao, M. D. *et al.* Streaming algorithms for identification of pathogens and antibiotic resistance potential from real-time MinIONTM sequencing. *GigaScience* **5**, 32 (2016). URL http://biorxiv.org/lookup/doi/10.1101/019356http://gigascience.biomedcentral.com/articles/10.1186/s13742-016-0137-2. DOI 10.1186/s13742-016-0137-2.

6. Cao, M. D. *et al.* Scaffolding and completing genome assemblies in real-time with nanopore sequencing. *Nature Communications* **8**, 14515 (2017). URL http://biorxiv.org/content/early/2016/05/22/054783.abstracthttp://biorxiv.org/lookup/doi/10.1101/054783http://www.nature.com/doifinder/10.1038/ncomms14515. DOI 10.1038/ncomms14515.

7. Quick, J. *et al.* Real-time, portable genome sequencing for Ebola surveillance. *Nature* **530**, 228–232 (2016). URL http://dx.doi.org/10.1038/nature16996http://www.nature.com/doifinder/10.1038/nature16996. DOI 10.1038/nature16996.

8. Faria, N. R. *et al.* Mobile real-time surveillance of zika virus in brazil. *Genome medicine* **8**, 97 (2016).

9. Quick, J. *et al.* Real-time, portable genome sequencing for ebola surveillance. *Nature* **530**, 228 (2016).

10. McIntyre, A. B. *et al.* Nanopore sequencing in microgravity. *npj Microgravity* **2**, 16035 (2016).

11. Castro-Wallace, S. L. *et al.* Nanopore dna sequencing and genome assembly on the international space station. *bioRxiv* 077651 (2016).

12. Boža, V., Brejová, B. & Vinař, T. Deepnano: Deep recurrent neural networks for base calling in minion nanopore reads. *PloS one* **12**, e0178751 (2017).

13. Stoiber, M. & Brown, J. Basecrawller: Streaming nanopore basecalling directly from raw signal. *bioRxiv* 133058 (2017).

14. 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*, 369–376 (ACM, 2006).

15. Jain, M. *et al.* Nanopore sequencing and assembly of a human genome with ultra-long reads. *bioRxiv* (2017). URL http://www.biorxiv.org/content/early/2017/04/20/128835. DOI 10.1101/128835. http://www.biorxiv.org/content/early/2017/04/20/128835.full.pdf.

16. Laszlo, A. H. *et al.* Decoding long nanopore sequencing reads of natural dna. *Nature biotechnology* **32**, 829–833 (2014).

17. David, M., Dursi, L. J., Yao, D., Boutros, P. C. & Simpson, J. T. Nanocall: an open source basecaller for oxford nanopore sequencing data. *Bioinformatics* **33**, 49–55 (2016).

18. LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. *Nature* **521**, 436–444 (2015).

19. Amodei, D. *et al.* Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, 173–182 (2016).

20. 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*, 770–778 (2016).

21. Ioffe, S. & Szegedy, C. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167* (2015).

22. Schuster, M. & Paliwal, K. K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* **45**, 2673–2681 (1997).

23. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural computation* **9**, 1735–1780 (1997).

24. Cooijmans, T., Ballas, N., Laurent, C., Gülçehre, Ç. & Courville, A. Recurrent batch normalization. *arXiv preprint arXiv:1603.09025* (2016).

25. Nair, V. & Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, 807–814 (2010).

26. Gers, F. A., Schmidhuber, J. & Cummins, F. Learning to forget: Continual prediction with lstm. *Neural Computation* **12**, 2451–2471 (2000). URL http://dx.doi.org/10.1162/089976600300015015. DOI 10.1162/089976600300015015. http://dx.doi.org/10.1162/089976600300015015.

27. Hochreiter, S. & Schmidhuber, J. Long short-term memory. *Neural Computation* **9**, 1735–1780 (1997). URL http://dx.doi.org/10.1162/neco.1997.9.8.1735. DOI 10.1162/neco.1997.9.8.1735. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

28. Kingma, D. & Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

29. Bankevich, A. *et al.* SPAdes: A New Genome Assembly Algorithm and Its Applications to Single-Cell Sequencing. *Journal of Computational Biology* **19**, 455–477 (2012). URL http://online.liebertpub.com/doi/abs/10.1089/cmb.2012.0021. DOI 10.1089/cmb.2012.0021.

30. Stoiber, M. H. *et al.* De novo identification of dna modifications enabled by genome-guided nanopore signal processing. *bioRxiv* 094672 (2017).

31. Sović, I. *et al.* Fast and sensitive mapping of nanopore sequencing reads with graphmap. *Nature communications* **7**, 11307 (2016).