Causalcall: Nanopore Basecalling Using a Temporal Convolutional Network

Abstract:

Nanopore sequencing is promising because of its long read length and high speed. During sequencing a strand of DNA/RNA moves through a biological nanopore, causing fluctuate in the current pore .Context-dependent current measurements are encoded into the DNA/RNA strand's base sequence during basecalling. For downstream studies like genome assembly and detecting single-nucleotide polymorphisms and genomic structural variations, accurate and quick basecalling is critical. Accurate basecalling remains a concern due to numerous variations in DNA/RNA molecules, noise during sequencing, and shortcomings of basecalling methods. In this paper, we propose Causalcall, which uses a deep learning model focused on end-to-end temporal convolution for accurate and quick nanopore basecalling.Causalcall, which is based on a temporal convolutional network (TCN) and a connectionist temporal classification decoder, detects base sequences of different lengths directly from current measurements in long time series.When compared to an RNN-based model, experiments on a variety of species have shown that the TCN-based model has a lot of potential for improving basecalling precision and time.Causalcall's effectiveness in reference-based genome assembly has also been shown in experiments.

Introduction:

Basecalling is a fundamental and pivotal step in nanopore sequencing technology. The basecaller is designed to identify the base sequences based on the raw current measurements. The development of basecallers can be roughly divided into two stages. Metrichor and Nanocall are representative basecallers from the first stage, which adopt hidden Markov model (HMM)-based methods. Both of them first convert current measurements to events (with each event corresponding to the movement of a k-mer through the pore). they implement an HMM and a Viterbi decoding algorithm to model the event space and further decode the base sequence.In this paper, we present Causalcall, which uses an end-to-end TCN-based model for nanopore basecalling. Causalcall is event-free and designed in the manner of sequence labeling. Therefore, it does not rely on precise label of each current measurement in training. During basecalling, Causalcall directly identifies base segments from the current measurements by using the TCN-based model, and assembles the resulting base segments to generate the final base sequence.TCN-based model is much simpler and more efficient.

- The main contributions of this paper can be summarized as follows:

● We modified the original architecture of TCN and combined it with a CTC decoder. The proposed model was able to accurately model the sequential features of long-range current measurements and classify base sequences of different lengths from a high-dimensional feature space.

● We demonstrated that the TCN-based model has the potential to improve the accuracy and speed of basecalling as compared to the RNN-based model. Causalcall achieves higher precision and is nearly three times faster than RNN-based Chiron in experiments on lambda phage, Escherichia coli, Klebsiella pneumoniae, and human samples.

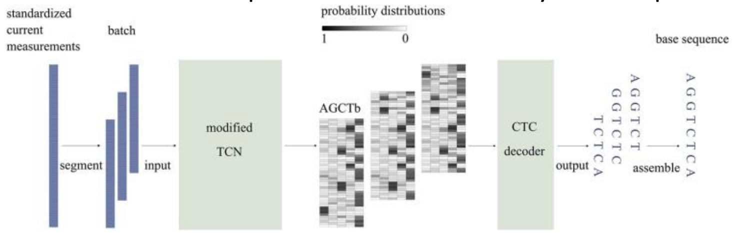
Method:

First we do data preparation, then Basecalling Workflow as shown:

The entire workflow of Causalcall is straightforward, without conversion of current measurements to events. The model is mainly developed on a modified TCN. Causalcall first standardizes the raw current measurements Xraw using median absolute deviation (MAD). The standardization formula is defined as:

$$X=c(Xraw-median(Xraw))median(|Xraw-median(Xraw)|)$$

A simplified procedure for determining the base sequence of a single sequencing read is shown in Figure. To increase speed, current measurements are first separated into T (T = 512) segments using a sliding window with a d (d = T/4) step size. The segments are then stacked in order and fed into the model in batches. Current measurements of a segment are modelled using a modified TCN and then mapped to probability distributions, which represent the probabilities of each base appearing at each time point, as the segment passes through the model. Based on the probability distributions, the CTC decoder identifies the base sequence corresponding to the segment at the conclusion of the model. Finally, Causalcall outputs the reconstructed sequence after assembling base sequences of segments from the same sequencing read by overlaps.The model's major purpose is to convert segments of current data with fixed length T into non-uniform length M (0 M T) base sequences. In this study, F symbolises the model, and Y = F (X) signifies the transformation, with X=[x1, x2,,xT], xiR and Y=[y1, y2,,yM], yj A, G, C, T, M T. A section of current measurements with T time points is denoted as X. A sequence of M bases is denoted by the letter Y. X and Y are sampled from the training set D = (X 1, Y 1), (X 2, Y 2),. throughout training. Causalcall learns the transformation from X to Y directly using an end-to-end deep neural network model. The main components of the model are a modified TCN and a CTC decoder. The improved TCN maps the present data' sequential properties to probability distributions of bases appearing at each time point. The CTC decoder then uses a beam search technique to find the most likely base sequence using probability distributions.



To construct a deeper architecture, the TCN stacks residual blocks. The original residual block was changed. Each block has two layers of stacked dilated causal convolution. The filter size (k), dilation factor (d), and number of filters are all the same in both stacked layers (n). Each dilated causal convolution layer receives weight normalisation, followed by a gated linear unit as the activation function. The gated linear unit's formula is H (H), where H stands for pointwise multiplication. The gated linear unit helps the model to choose which features are important for correctly predicting the bases. After that, a residual connection is applied to the second convolution layer's activated output and the block's input, followed by a ReLU activation function. The receptive field of convolution for the fifth block is 32. It's enough to cover the present measurements that are related to the pore's base shifting outward. To map current measurements to feature space, the stacked residual blocks operate as a feature extractor. The retrieved features represent the correlation of the present measurements at different time points because the causal convolution is implemented in the time dimension. The last residual block is then followed by two FC layers, with a softmax function added at the end. FC layers are used to combine features from various channels. The output of the last FC layer is then transformed into a matrix of probabilities, with each row representing the likelihood of bases existing at that time point.

Metrics for Basecalling Evaluation

We linked basecalling data to the reference genome to assess basecaller performance. First, using minimap2 and the map-ont option, the base sequences from various basecallers are mapped to the reference genome. Mapping data are saved in SAM files and examined using the japsa1 error analysis programme (jsa.hts.errorAnalysis) to determine the identity and error rates of each mapping, including insertions, deletions, and mismatches. Correct alignments between the base sequence and the reference genome are indicated by identity. The identity rate is calculated as the number of matched bases divided by the total number of bases in the reference. It's a statistic for basecalling precision. The numbers of inserted bases, deleted bases, and mismatched bases divided by the number of bases in the reference genome yield the insertion rate, deletion rate, and mismatch rate, respectively. Incorrect alignments are indicated by mismatch. The term "loss/insertion" refers to a base deletion or insertion in the base sequence compared to the reference genome. They are basecalling error metrics. The total error rate was calculated by adding the rates of insertions, deletions, and mismatches. Furthermore, basecalling speed is an essential criterion for evaluating basecallers' performance. The amount of outputted bases divided by the running duration yields the speed. Higher identification rate and speed indicate greater performance, but lower values indicate better performance for the other parameters.

Genome Assembly and Error Analysis

We also used the assembled genome's quality to compare readings from different basecallers. The base sequences of K. pneumoniae are constructed in this study utilising Rebaler's assembly method. K. pneumoniae samples have no plasmids and, on average, a sequencing depth of over 100, making them easier to assemble. There are two primary steps in the assembling pipeline. To build a draught assembled genome, the matching portions in the reference genome are replaced with read sequences based on the mapping results of minimap2. Second, 10 rounds of Racon are used to polish the draught assembled genome. The workflow makes sure that the assembled genome and the reference genome are roughly the same size. The completed genome is first divided into 10 kbp segments for effective mapping. Then, using minimap2 and the axe asm option, these bits are mapped to the reference genome. The mapping findings are used to calculate the rates of identity, deletion, insertion, and mismatch. Base deletion and substitution can be caused by a variety of causes, including DNA homopolymerization and methylation. In the context of the reference genome, mapping mistakes in assembled genomes are further divided into six groups based on their impact. The error analysis method described in Rebaler is implemented using the assembled genomes of K. pneumoniae. The nucmer command in MUMmer v4.0.0 is used to map pieces of the assembled genome to the reference genome. The delta-filter and show-snps commands are then used to find single-nucleotide polymorphisms (SNP). Errors are characterised as homopolymer deletion, homopolymer insertion, Dcm error, insertion, deletion, and substitution based on the results of mapping and SNP detection. Homopolymer insertion and homopolymer deletion are insertion and deletion errors that occur in homopolymer areas with more than two bases, respectively. Dcm error represents errors that occur in Dcm-methylation motifs (CCTGG or CCAGG). Errors not in the former three categories are classified as ordinary deletion, insertion, or substitution

Training:

different validation set to prevent overfitting. The existing measures are standardised using MAD and separated into segments of a given length prior to training (as described in the Data Preparation section). Batches of segments are input into the model. We employ a 512-segment length and a 256-batch size. The training set and validation set are mixed after each epoch during training. As the number of iterations grows, the learning rate slows down. The model is trained on an NVIDIA 1080ti GPU with 12 GB of RAM, and it takes approximately three days to coverage.

Related work:

The basecaller is designed to identify the base sequences based on the raw current measurements.

● Basecaller architecture can be divided into two phases. Metrichor and Nanocall are examples of first-stage basecallers that use hidden Markov model (HMM)-based methods. Both of them translate current

measurements into events (each event corresponding to a k-mer passing through the pore). Then, to model the event space and decode the base sequence, they use an HMM and a Viterbi decoding algorithm.Metrichor is the first basecaller provided by Oxford Nanopore Technologies (ONT), and it works in the cloud. Nanocall is an open-source and third-party basecaller. It works offline, but only on data generated with pore version R7.3 signal measurements.
(https://doi.org/10.1093/bioinformatics/btw569)

● In the second stage of basecaller development deep learning-based approaches became popular for basecalling. An example of these is Deepnano, models statistical characterizations of events and then predicts base sequences using a bidirectional recurrent neural network (RNN). For the R7.3 platform, it outperforms Metrichor and demonstrates RNN's ability in basecalling.
(https://doi.org/10.1371/journal.pone.0178751)

Result:

On the four species, Causalcall had a greater identity rate and a lower mistake rate than Chiron. Causalcall has lower insertion and mismatch rates in terms of three mapping mistakes. Causalcall, in particular, has the lowest phage and human insertion rate. These findings demonstrate that Causalcall was successful in learning the base-related patterns hidden in the current data. Furthermore, they show that the Causalcall TCN-based model has a better chance of improving basecalling accuracy than the Chiron RNN-based model. Causalcall is roughly three times quicker than Chiron when it comes to basecalling speed, averaging 7,000 bases per second. This increase in speed implies that the TCN-based approach, which is based on the matrix computing of dilated causal convolution, does indeed speed up basecalling. In addition, when compared to Chiron, Causalcall uses longer chunks of current measurements with a lower overlap ratio. This also aids in the acceleration of the process. Guppy and Flappie's training data are not publically available, as we discussed in the preceding section. As a result, it's difficult to establish whether the strong performance is due to the training data or their RNN-based models. However, we discovered that the training data does have a considerable impact on basecaller performance. On average, Chiron (DNAre) has a 1.78 percent higher identity rate and a 1.85 percent lower error rate than Chiron (DNAde), confirming the training data effect. Flappie is written in the C programming language and is accelerated by several CPU threads. Flappie's speed is determined by the number of threads and the CPU's current condition. As a result, the speed may vary substantially. Causalcall is as fast as Flappie with 20 CPU threads, according to test findings. Guppy is a binary version of a produced software. Because it is not open source, we don't know what software engineering technologies it uses for acceleration. Guppy is a developed software in binary version. We do not know its inner software engineering technologies used for acceleration, as it is not open source. Guppy works much faster than Causalcall at present, but Causalcall can be further accelerated by reprogramming with C language and integrating with acceleration technologies such as parallel computing. The results above demonstrate that the TCN-based model of Causalcall could improve the accuracy and speed of basecalling.