Basecalling Raw Nanopore DNA Sequencing Reads using Neural Networks

Máté Borkó, Bence Bolgár, Peter Sarkozy
Department of Measurement and Information Systems
Budapest University of Technology and Economics
Budapest, Hungary
borkomate@gmail.com

***Abstract*—The single-molecule real-time (SMRT) DNA sequencer developed by Oxford Nanopore Technologies offers breakthrough read lengths in a handheld device, while greatly simplifying input DNA library preparation procedures. The standard method of identifying the individual bases passing through each pore relies on a Hidden Markov Model (HMM), mapping raw current levels to individual bases in a nonlinear, multi-staged fashion. Recent advancements in artificial neural networks (ANN) and related natural language processing (NLP) techniques allow novel neural architectures that may improve the accuracy of the basecalling. We developed and examined a novel deep neural network based method to perform basecalling on raw current level measurements, as well as an efficient method of selecting and curating a training database from a set of real measurements.**

***Index Terms*—DNA sequencing, neural network, LSTM, Nanopore**

I. Introduction

Oxford Nanopore Technologies's (ONT) MinION device uses a new single-molecule real-time technique to read individual DNA sequencing. The greatest achievement of this effort is the ability to sequence a single molecule of DNA without resorting to enzymatic amplification of the strand. ONT makes unprecedented read lengths possible, up to tens of thousands of bases long, compared to several hundred bases long provided by current technologies. One disadvantage of this new platform is that the basecalling accuracy (the ratio of correctly identified nucleotides in a sequence) is an order of magnitude lower than other competing platforms. Sequencing is performed by passing each individual single stranded DNA molecule through an engineered nano-scale pore, along with a ratcheting helicase enzyme to slow the passing of the molecule to levels where the characteristic current of the nucleotides inside each pore can be captured. The current is sampled at 4 kHz, while the average traversal rate of the DNA strand is approximately 300-500 nucleotides per second. The current level is representative of multiple nucleotides (up to 6), thus currently Hidden Markov Models are used to perform the basecalling.

The use of neural networks in the field of DNA sequencing is a recent phenomenon, with only a few existing solutions [3]. The performance of deep neural networks in natural language processing (NLP) gave us a good starting point, as they share a strikingly similar underlying model from a signal processing aspect. We aim to establish a high quality training set for future neural network basecallers, and to identify an efficient and NN architecture for basecalling. A successful neural network basecaller architecture requires two vital components:

- The accuracy of any neural network is inherently limited by the amount and quality of available training samples.
- An appropriate neural network structure must be declared. The number of layers and neurons must be selected according to the required result, and is governed by a suitable cost function which allows the optimization of the structure with respect to training time and accuracy.

II. Creating Training Samples

A large, publicly available dataset, in the form of the whole genome shotgun sequencing of the NA12878 human sample [4] was used as a training set. This multi-TB dataset contains the entire human genome at approximately 30x coverage, and provides a wide variety of training data. We used the GRCh38 reference human genome as a gold standard, as its deviation from the true NA12878 is orders of magnitude lower than our expected error rates.

A. Original Mapping Procedure

Initially, we attempted to implement a method similar to how Metrichor [5], the vendor-released basecaller maps input signals to DNA sequences. This 3-step method functions as follows:

- The raw data files containing the time-current measurements are separated into contiguous segments with no changes in current called events.
- The basecaller assigns a k-mer probability to each event, using a HMM.
- The kmer probabilities along with the HMM step and stay probabilities are calculated to assign a final basecalled sequence and a PHRED score

The CIGAR string describes the alignment between the read and the reference sequence, noting soft clipped (S), matched (M), inserted (I) and deleted (D) bases. The process is summarized on Fig. 1.

B. Sorting Training Data

Using the previously described method we were able to assign raw data samples to reference bases. Initially, we

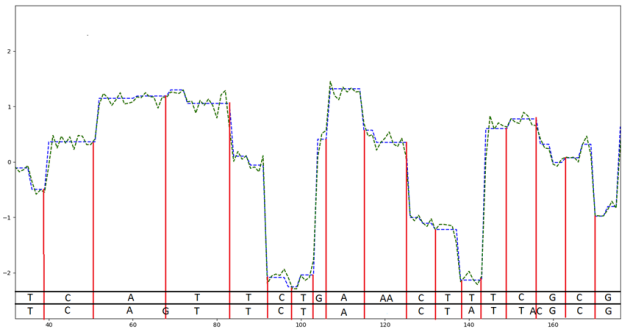

Fig. 1. *Abstraction levels during basecalling. Raw current levels are shown in green, event current levels in blue. Red columns denote each event start. The vertical axis shows raw normalized current, and the horizontal axis shows the reference sequence (not time scaled).*

attempted training the neural network directly with these segmented signals, but further research suggested an architectural element, Connectionist Temporal Classification (CTC) [10] that made data segmentation unnecessary.

All raw data files contained an offset to the start of the segmentation, but the duration of some events were not defined correctly, so the raw data-to-event mapping could not be executed trivially. We corrected the durations by calculating them from the start and stop times of each event. The results of each measurement are stored in $fast$ files. These files contain all relevant measurement related metadata, e.g. the event segmentation, offset, duration, temperature, and environmental variables that could be transferred into the neural network model. We extracted the $fastq$ sequences from the $fast5$ files, and mapped them to the reference sequence using the Burrows-Wheeler Aligner (BWA, [6]) and Samtools [7]. The aligner reports quality information about each alignment, which we used to identify candidate reads for use as training samples. Sequences that aligned to the positive strand of the genome with a mapping quality threshold above 30 were used as training samples. Additionally, if the CIGAR string of an alignment indicated the presence of deletion more than 5 bases long, the read was discarded from the training set [8].

C. Final training set

During the first phase of our research, we examined about 300 GB of raw measument data, originating from approximately 40000 reads. We initially selected 160 MB, or 1971 training sequences, to be representative of the entire set. The training data consists of raw current signals and reference nucleotide sequence pairs as inputs and outputs, respectively. These files were sorted by length, as zero padding of shorter sequences is required for efficient mini-batch training [12]. A neural network trained with these samples is capable of translating raw current signals into nucleotide sequences, as per the goal of our research. We normalized the samples before storage.

III. Selecting the Appropriate Architecture

Recent research has created new artificial neuron models, where the output of a neuron is the outcome of a more complex process. In this case the output depends on the last n (hidden)states of the neuron and on the last input sample, so an output at the time point t_i carries the effect of all previous inputs indirectly. A networks built from such elements are called Recurrent Neural Networks (RNN) [13]. The stored n states enables the handling of the time dependencies in the input sequence.

A. LSTM

During the backpropagation (training) RNN cells cannot prevent exponentially increasing gradients. The errors have to be backpropagated through all hidden neurons (states), which is not possible, if the gradients grow to the infinity. As a result, RNN structures allow only a finite number of hidden states, thus the time dependency that can be handled by RNNs is shorter than what is required for basecalling.

Long-Short Term Memory is an architecture that can handle such long distance time dependencies [9]. In this model the neurons are replaced with small memory cells, and these cells have control over the output. Through this control they can prevent the problem of exponentially increasing gradients.

The functionality of these cells can be described with 6 variables:

- g is the input
- i is the input gate. Its value is multiplied by the input, so it functions as a weight.
- f is the forget gate. It controls the influence of the hidden state.
- o is the output gate. It weights the value of the output
- s represents the state of the cell.
- h symbolizes the output.

Furthermore W are the related weight matrices and b the biases for every state variable. The activation function, represented by σ is a $tanh$ function. The equations that describe the functionality of an LSTM cell:

$$\begin{aligned} g_t &= \sigma(W^{gx}x_t + W^{gh}h_{t-1} + b_g) \\ i_t &= \sigma(W^{ix}x_t + W^{ih}h_{t-1} + b_i) \\ f_t &= \sigma(W^{fx}x_t + W^{fh}h_{t-1} + b_f) \\ o_t &= \sigma(W^{ox}x_t + W^{oh}h_{t-1} + b_o) \\ s_t &= g_t \cdot i_t + s_{t-1} \cdot f_t \\ h_t &= \sigma(s_t) \cdot o_t \end{aligned} \tag{1}$$

B. Bidirectional Networks

Alongside LSTM, bidirectional neural networks [11] have become the most important development of RNN structures, as these have been used to recognize handwritten letters. Bidirectional cells contain two hidden neurons that are in connection with the input cells and also with the output cells. One of them has a recurrent signal from the input layer and the other one from the output layer. This structure can learn the

sequence from the both directions, and predict the output as the outcome of the previous and next time samples. Naturally, this functionality enables only offline processing.

C. CTC

The core of the processing model is the CTC cost function [10]. The usage of CTC makes input data segmentation unnecessary in the preprocessing phase. CTC cost functions with bidirectional LSTM structures have outperformed Hidden Markov Models in many applications [10].

The brief functional description of CTC is as follows: Alongside the standard output labels (here A, G, C, T) CTC introduces a "blank" label (b), and fills the desired output sequence with these blank labels. E.g. the sequence "$ACGT$" would be "$bAbCbGbTb$". This is the extended output. In next step, a two dimensional graph is declared, which contains a number of nodes equal to the input sequence length in the horizontal directions and extended output length number nodes in the vertical direction. The applied neural network predicts the probability of each label (A, C, G, T, b) in every time step. There are permitted and prohibited transitions regarding the output sequence. E.g. in the sequence "$bAbCbGbTb$" the prediction can start only with b or A and if a A has been predicted, the next predictions can be A or b or C, but G can not follow A, because it would cause loss (C would be eliminated from the output). Similar to Hidden Markov Models, two types of variables are used to model the state of the predicted probabilities. A forward and a backward variable for the forward and backward traversal of the graph. Finally, the blank labels are removed from the predicted output sequence, and depending on the decoder style, the repeated labels will be collapsed.

IV. Results

We initially started the formulation of the neural network architecture without any specific restraint, and created models 1-2 hidden layers with a number of neurons between 50 and 1000. We used a single test sample to configure the initial model structure, and the length of the input (raw current level) at 9000 samples caused two main issues: a single training iteration took longer than two minutes, and the CTC architecture required the entire input sequence, which taxed the memory subsystem of our hardware. We only used input sequences that were shorter than 4000 samples in the current domain to decrease training times. In order to select the optimal number of neurons in each layer, we analyzed the relationship between the raw current vector length and the output nucleotide sequence. We used the reference sequence to determine that each single nucleotide corresponds to approximately 11 raw current samples. However, the effect of a single nucleotide on the current inside of each pore extends further, to approximately 20 nucleotides [14]. This results in approximately 220 raw current samples, and thus 200 neurons were chosen to the output per time step. This network used a mixture of LSTM and bidirectional neural networks with 100-100 neurons in the forward and backward layers.

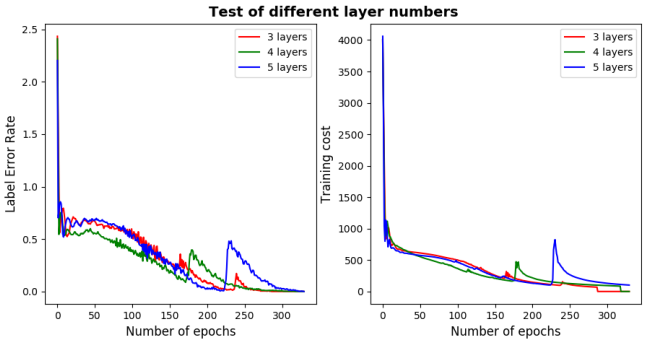

Fig. 2. *Training with different layer sizes*

Networks with only 1 or 2 layers were unable to learn the training data, so we increased the depth of our model. The results of 3, 4, 5 layers are shown in the Tab. I and Fig. 2.

TABLE I
Layer test results

Number of layers	Epochs	Training time [s]
3	287	2348.37
4	318	3681.05
5	330	4743.70

We determined that using 3 layers, the accuracy of the network depended highly on the initially randomized weights, while with 5 layers, the training time was unnecessarily long. Thus a 4 layer model was chosen. Further results all used 4 layers with 200 neurons per layer structures with the ADAM optimizer and CTC loss function. The performance on unidirectional networks were also investigated, so we compared the reference network with a 4 layer feedforward neural network using 200 neurons per layer. As the Fig. 3 shows it performed markedly worse. It was unable to learn the reference sequence in 1000 iterations. Thus we concluded that unidirectional networks do not have sufficient performance for applying them in basecalling raw nanopore current levels.

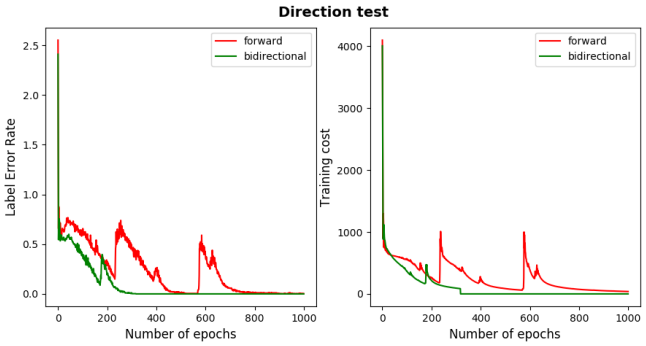

Fig. 3. *Comparison of forward and bidirectional networks*

To enhance the power of our network we implemented almost all of the optimization opportunities provided by the Tensorflow framework. We used features such as dropout training, using mini batches, shuffling the order of the training

data and layer normalization. Dropout and shuffling the order of training data help avoid getting stuck in local minima and overfitting. Mini batches aid in more robust convergence. The optimal batch size is yet to be determined, as we currently use one sample per batch (batch size 1). Finally, layer normalization speeds up convergence and improves the achieved accuracy. All implementations used the shuffling of the order of training data. The Fig. 4 shows the result of dropout training.

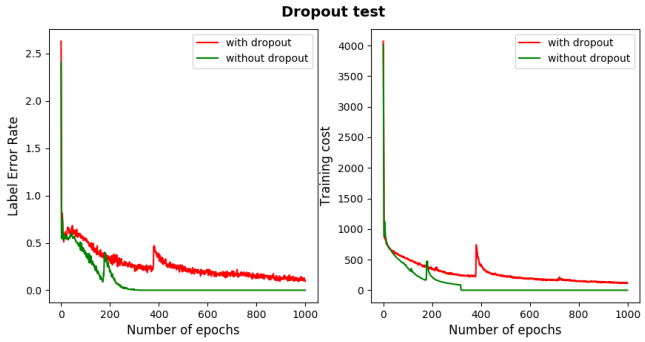

Fig. 4. *Dropout test*

As it is shown in the Fig. 4, we experienced, that the dropout stabilizes the performance of the network, even so we used it in further models. We initially had 34 training samples with the input lengths shorter than 4000. The first acceptable network was trained on these samples. The hyperparameters: 4 bidirectional layers with 100-100 neurons, dropout 0.5, learning rate 0.003. The training samples were divided into two parts, training samples (27) and test samples (7). The results are shown Tab. II. During the training we got 5% error rate on the training samples. We analyzed a larger subset of the entire raw dataset to collect more samples meeting our criteria. This increased our training set to 79 sequences shorter than 4000 samples, and were divided into training and test data by a ratio of 64/15. The structure of the network was not changed. The results in Tab. II show an accuracy of 4.8% on the updated training samples.

TABLE II
TRAINING RESULTS

Number of training samples	27	64
Number of test samples	7	15
Average match rate	0.617	0.679
Average mismatch rate	0.112	0.084
Average insertion rate	0.198	0.134
Average deletion rate	0.072	0.104
Average label error rate	0.383	0.321
Training time	7 days 6 h	13 days

With respect to the label error rate, this nets an additional 5% improvement, resulting in a 68% final accuracy. While this accuracy is lower than the best currently available basecallers (up to 90%), it is still a promising result considering the small number of training samples.

V. CONCLUSION

Although our model has not yet reached the accuracy of Albacore yet, we have not found any reason why it could not be achieved. The reached 68% is not a record-breaker, but we must mention that the number of training samples used is extremely low. Our only major hurdle appears to be very long network training time, and we must explore further options to decrease it. To further increase the accuracy of our model, we plan to use longer training sequences. Adding longer sequences to the network directly has dire consequences in terms of memory usage and increased training times. Thus we plan to segment the data into overlapping partitions with a sliding window, allowing the use of arbitrary length raw sequences as training data. We also plan to investigate the addition of a convolutional layer into the model architecture, as such a mixture of layers often give excellent results in practice [10].

VI. ACKNOWLEDGEMENTS

The authors acknowledge that they are participants in the Oxford Nanopore Technologies's MinION Access Program, and have no other conflicts of interest. This research was supported by the OTKA-K-112915 Grant, and the Multipurpose Health Monitoring Platform bilateral Croatian-Hungarian grant. The Titan Xp used for this research was donated by the NVIDIA Corporation.

REFERENCES

[1] Oxford Nanopore Technologies. Electronics for nanopore sensing. https://nanoporetech.com/how-it-works 05-20-2017
[2] Mikheyev AS., Tin MM., A first look at the Oxford Na- nopore MinION sequencer. Mol Ecol Resour 14(6): p. 1097-102. DOI 10.1111/1755-0998.12324 2014
[3] Wick, RR., Holt, KE., et al.: Comparison of Oxford Nanopore basecalling tools. https://github.com/rrwick Basecalling-comparison 11-09-2017
[4] Nanopore Whole Genome Sequencing Consortium, https://github.com/nanopore-wgs-consortium/NA12878 11-15-2017
[5] Oxford Nanopore Technologies, https://nanoporetech.com/ 2017-12-20
[6] Li H., Durbin R.: Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics, 26, 589-595. 2010
[7] Li H., et al: 1000 Genome Project Data Processing Subgroup (2009) The Sequence alignment/map (SAM) format and SAMtools. Bioinformatics, 25, 2078-9. [PMID: 19505943]
[8] Sarkozy P., Antal, P., Jobbágy, Á.: Calling Homopolymer Stretches from Raw Nanopore Reads by Analyzing k-mer Dwell Times. 2016. DOI: 10.1007/978-981-10-5122-7-61.
[9] Schmidhuber, J., Hochriter, S.: Long Short-term Memory. 1997. http://www.bioinf.jku.at/publications/older/2604.pdf, doi: 10.1162/neco.1997.9.8.1735
[10] Graves, A.: Supervised Sequence Labelling with Recurrent Neural Networks, chapter 7. Connectionist Temporal Classification. 2012. doi: https://doi.org/10.1007/978-3-642-24797-2.
[11] Schuster M., Paliwal, KK., Bidirectional Recurrent Neural Networks , IEEE TRANSACTIONS ON SIGNAL PROCESSING, VOL. 45, NO. 11, NOVEMBER 1997, doi: 10.1109/78.650093
[12] Bottou, L.: Large-scale machine learning with stochastic gradient descent. In COMPSTAT2010 (pp. 177-186). Physica-Verlag HD. 2010
[13] Hopfield, J. J. (1982). Neural Networks and Physical Systems with Emergent Collective Computational abilities. Proc. Natl. Acad. Sci. USA,79, 2554-2558.
[14] Teng, H., et. al.: Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning, doi: https://doi.org/10.1101/179531