Sequence Analysis

Calling homopolymers in nanopore sequencing data

Marijke M.A. Thijssen*

[1]Department of Bioinformatics, Wageningen University and Research, 6708PB, Netherlands

*To whom correspondence should be addressed.

Received on February 28, 2019

Abstract

Motivation: Nanopore sequencing is a fast developing sequencing technology capable of producing long DNA reads in real-time. The generation of long reads helps solving obstacles in whole genome analysis that the common short reads are unable of. However, low basecalling accuracy in homopolymer regions constrains the potential of the technology. A number of tools exist to correct such errors, however these require substantial computational power. Here, we propose a new approach to homopolymer calling. By selecting only those regions that actually contain homopolymers prior to basecalling, the investment of computational resources for homopolymer correction can be reduced.

Results: We trained two different neural network architectures to detect homopolymer stretches in the raw nanopore signal generated by a MinION sequencer, the gated recurrent unit recurrent neural network (RNN) and the hybrid residual network-RNN (ResNetRNN). ResNetRNNs showed a greater capacity to detect homopolymers than RNNs. The best performing network accurately detected two thirds of the homopolymers present in the raw signal, but also overestimated the actual number of homopolymers. We incorporated this network in a pre-processing tool that splits the nanopore signal on homopolymer content for specialised treatment. We highlight several venues to further improve the model and argue that our tool, combined with specialised basecallers, can reduce the computational cost and error in basecalling nanopore reads in the future.

Availability: All code used in this research is freely available at https://git.wur.nl/thijs030/thesis. The tool is freely available at https://git.wur.nl/thijs030/thesis/tree/master/catfish.

Contact: marijke.thijssen@wur.nl

1 Introduction

We are currently moving towards the stage of third-generation sequencing (TGS): sequencing of single molecules, omitting the need for amplification or interruptions for base incorporation (Heather & Chain, 2016). TGS technologies are maturing rapidly and pose great advantages over the current golden standard second-generation sequencing (SGS) technologies: real-time sequencing, reduced cost per instrument run, detection of epigenetic modifications and generation of long reads (Rang *et al*., 2018).

One of the major TGS techniques is nanopore sequencing by Oxford Nanopore Technologies (ONT). A single-stranded DNA molecule is translocated through a nanopore over which an electrical potential is applied. The current is measured as the strand moves through the nanopore. This raw signal is translated into a DNA sequence by so-called basecallers. This is a challenging process, as multiple (on average 5) bases simultaneously influence the signal, while moving through the pore at a non-uniform speed (Agah *et al*., 2016). Nanopore sequencing has no theoretical read length limit and reads usually exceed 10 kb, as opposed to the short reads of up to 300 bp generated by SGS technologies. Reads with lengths over 2 Mb, so-called ultra-long reads, have even been reported (Payne *et al*., 2018).

In 2014, ONT introduced the portable MinION nanopore sequencing device. The small and light-weight device has a low initial cost of $1,000.- (Oxford Nanopore Technologies, 2018). The platform has the promise of high accessibility to sequencing, even in resource-limited setting. The device connects via a USB port to a PC or laptop. No expensive hardware, continuous power or trained personnel is needed (Elliott *et al*., 2018). The MinION sequencer was successfully exploited in monitoring the Ebola virus during the outbreak in 2015 (Quick *et al*., 2016). This illustrates the practical application of nanopore sequencing.

Nanopore sequencing suffers from a low accuracy of about 90% to 98%, compared to SGS methods that have accuracies over 99% (Oxford Nanopore Technologies, 2017; Sárközy *et al*., 2018). SGS technologies are able of cheaply producing *de novo* assemblies, however repetitive

regions cause fragmentation, gaps and ambiguity in the genome that SGS technologies cannot resolve. Long reads can span over these repetitive regions and close the genome. Generating long reads helps bridge problems in analysis of whole genomes, haplotype phasing and structural variant detection, among others (Dijk van *et al*., 2018; Lu *et al*., 2016; Sedlazeck *et al*., 2018). Since the error rate in nanopore sequencing ranges from 2% to 10% at the moment, there is a need for improvement in accuracy to reach the methods' full potential.

A major issue in nanopore sequencing is the basecalling of homopolymers, here defined as a stretch of five or more identical bases. Basecallers including Albacore[1] (Boža *et al*., 2017; Teng *et al*., 2018) collapse homopolymers into shorter stretches if the homopolymer length exceeds the number of bases simultaneously influencing the measured current, resulting in a high deletion rate in nanopore reads (Jain et al., 2018). Currently, basecalled reads are often assembled into a consensus sequence after which by so-called polishers such as Nanopolish (Simpson *et al*., 2017) map the reads back to the assembly to improve the consensus. The consensus accuracy after polishing is close to 99.7% in general (Wick et al., 2018), but it is a computationally expensive process. Current basecallers are not specialised to detect homopolymers. ONT is working on new pores that allow for more accurately measuring the bases as they pass through (Brown, 2018). However, for now, a significant gain could be achieved if the problem in calling of homopolymers can be addressed.

Here, it is proposed to select homopolymer stretches for specialised treatment. By selecting only those regions that actually contain homopolymers for correction, the investment of computational resources and time can be reduced. Due to the complex and noisy nature of nanopore sequencing data, neural networks were applied with the goal to detect homopolymers from non-homopolymers in the raw signal generated by a MinION nanopore sequencing device. In this research, we developed and evaluated two different neural network architectures for this purpose. One network was further developed to be incorporated into the homopolymer selection tool *catfish*, that pre-processes the reads with the ultimate goal to improve nanopore sequencing accuracy by correctly calling homopolymers.

2 Methods

2.1 Tool design

The tool catfish was designed as part of the workflow to improve basecalling and speed up the process (**Figure 1**). Catfish takes in the directory to the raw nanopore signals saved in FAST5 format and the name for an output directory. The reads are first trimmed and median normalised. Next, a neural network at the core of catfish classifies the measurements making up the raw signal as part of a homopolymer or not. This output is corrected to remove consecutive homopolymer measurements that are shorter than five times the minimal event length of three measurements, as this is the expected minimal true homopolymer length. The signal is split on these homopolymer regions and saved to separate FAST5 files. The names of the split files are returned by catfish. The workflow continues by basecalling nanopore reads containing no predicted homopolymers speedily using Albacore 2.3.3 with default settings. Reads that do contain assumed homopolymers, are basecalled using Albacore 2.3.3 with experimentally found settings optimal for homopolymer basecalling.

2.2 Neural network design

As nanopore sequencing data are complex and noisy, likely to have highly non-linear relations, recognising patterns within that data is a task well suited for machine learning techniques. Neural networks are known for their capability to recognise complex patterns and to generalise to unseen data. Several neural network architectures have been designed and are applied to a variety of problems. Convolutional neural networks (CNNs) have been applied to ONT data before for demultiplexing (Wick *et al*., 2018) and protein sensing (Misiunas *et al*., 2018). Bidirectional recurrent neural networks (RNNs) (Schuster & Paliwal, 1997) have

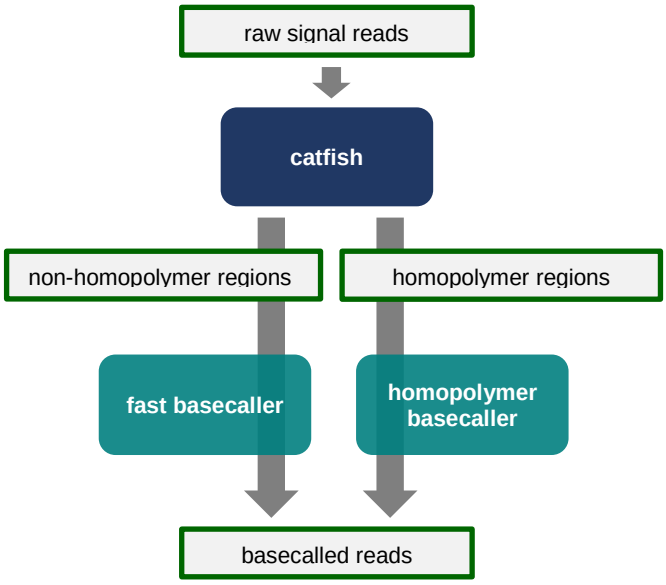

Figure 1 - Workflow. The raw signal generated by a MinION sequencer is classified by the tool. Reads are split on regions that contain homopolymers in separate FAST5 files. Files containing no homopolymers are basecalled by a fast basecaller. Files containing a homopolymer are basecalled by a specialized basecaller.

[1] https://community.nanoporetech.com/protocols/albacore-offline-basecalli/v/abec_2003_v1_revan_29nov2016

been used before in ONT basecalling (Boža et al., 2017), as well as a CNN combined with a RNN (Teng *et al*., 2018). Basecallers DeepNano and Chiron both employ the Long Short-Term Memory (LSTM) unit (Hochreiter & Schmidhuber, 1997), a more advanced RNN unit. Common difficulties encountered in training neural networks are the vanishing and exploding gradient problems, especially in deep networks (Hochreiter *et al.*, 2001). Networks that suffer from these unstable gradient problems cannot efficiently tune the parameters because the updates are too small or large. A number of solutions exist to diminish or solve this problem.

Initially, the bidirectional gated recurrent unit RNN (biGRU-RNN) architecture was employed in this research. RNNs make use of dependencies inside the data and are fit for sequential data. The GRU-RNN (Cho *et al*., 2014) is an improved adaptation of the standard RNN, which solves the vanishing or exploding gradient problem by making use of a gating mechanism (Chung *et al*., 2014) (**Supplementary Figure 1**). Because the prediction of a homopolymer can be influenced by both information before and after the measurement, a bidirectional version of the RNN was implemented. Secondly, a more intricate and potentially more powerful architecture was adopted, namely a combination of the aforementioned RNN and a ResNet (ResNet-biGRU-RNN, further referred to as ResNetRNN). A ResNet (He *et al*., 2016) is a specialized type of CNN (LeCun *et al*., 1999) that does not suffer from the vanishing or exploding gradient problem because of skip connections inside the network (**Supplementary Figure 2**). This allows for building deeper networks, which have a greater capacity for feature representation. CNNs have proven to be excellent local feature extractors and have been successfully applied to ordered one-dimensional data including text (Kim, 2014) and sensor data (Rueda *et al*., 2018). The hybrid architecture potentially benefits from the advantages of both types of network by combining the different methods of feature extraction.

Both types of networks were connected to a final fully connected layer that produces the final output. The networks were built in TensorFlow (Abadi *et al*., 2016) in Python 3.5. Network weights were initialised using Xavier initialisation to avoid starting weights in saturated zones, which could result in a vanishing or exploding gradient (Glorot & Bengio, 2010). The ReLU activation function, robust to the vanishing gradient problem and easier to train, was applied in the ResNet layers. GRUs were used that have a tanh activation function for candidate memory determination and sigmoid activation functions for gating (**Supplementary Figure 1**). Biases in GRUs were initialised to one. Additionally, dropout was applied to the RNN layers to prevent overfitting (Srivastava *et al*., 2014). Batch normalisation was applied in the ResNet layers to accelerate training and improve regularisation (Ioffe & Szegedy, 2015). The probability for a homopolymer was calculated using the sigmoid cross entropy function.

2.3 Hyperparameter selection

The appropriate hyperparameters, which are variables that are initialised before training, are crucial to the success of the network as they significantly impact the behaviour of the model. However, the ideal combination of hyperparameters cannot be known in advance. Therefore, random search was applied to search a large hyperparameter space for hyperparameter optimisation within the available computational budget and time (Bergstra & Bengio, 2012). The hyperparameters learning rate, optimiser, size of layers, mini-batch size, number of layers and dropout rate were varied (see **Supplementary Table 1** for a description of the hyperparameters; see **Supplementary Table 2** for an overview of ranges). A total of 32 RNNs were generated for random search.

To assess if ResNetRNNs have a greater capacity of learning useful features in this problem, a two-fold approach was operated on. Firstly, fifteen ResNetRNNs were randomly generated as part of the random search, as there is a fair possibility that the ideal set of hyperparameters differs per network architecture. Secondly, the hyperparameter settings of the top four performing RNNs were reused as settings for the RNN substructure and combined with a ResNet of which the hyperparameters were randomly selected. It was hypothesized that a well-performing RNN is a strong basis for a ResNetRNN. For each of the four best performing RNNs, five new instances of a ResNetRNNs were created. A total of 35 ResNetRNNs were constructed.

2.4 Data preparation

The data set used in this research was a genomic, MinION generated set on *Escherichia coli* K-12 MG1655, made publicly available by the Loman research group (Loman, 2017). Samples were sequenced with the ONT 1D Rapid Sequencing Kit SQK-RAD002 on a standard FLO-MIN106 R9.4 flow cell using MinKNOW1.4. The nanopore reads were basecalled by Albacore 0.8.4. Data were saved to a specialized hierarchical data format 5 (HDF5) file, FAST5, which contains both the raw signal and metadata such as basecalling information.

The data were median normalised and corrected using the tool Tombo (Stoiber *et al*., 2016) resquiggle with four threads, which assigned the raw signal to the given reference (Blattner & Plunkett, 2014) based on the basecalled sequence and current, which resulted in a correction of some errors compared to the reference. Reads that could not be processed by Tombo were omitted. Non-aligning bases at the start and the end of the reads were trimmed off. After this procedure, the data set had a homopolymer content of

2.22%, matching that of the *E. coli* genome (**Supplementary Table 3**).

The measurements in the raw signal were labelled as being part of a homopolymer or not according to the underlying base sequence (**Supplementary Figure 3**). A homopolymer is here defined as a stretch of five equal bases or longer. Measurements were linked to bases via events, segments of the raw signal that correspond to an unchanging set of nucleotides occupying the pore. If a measurement was part of an event that points to a base that is part of a homopolymer, the measurement is labelled as being part of a homopolymer. The complete read set of 81,703 reads was randomly distributed over a training, validation and test set of 70%, 15% and 15% of all reads, which corresponded to 51,792, 12,255 and 12,256 reads respectively.

2.5 Building a database

As the purpose of the network is to accurately detect homopolymers, using a balanced training set will create the possibility for the network to learn the patterns for both the minority and majority group (Buda *et al*., 2018). As homopolymers are severely less present in the *E. coli* genome, a database was constructed to hold homopolymer and non-homopolymer examples, so a balanced set of training examples could be extracted for training. Each example was composed of a randomly selected stretch of 35 measurements in length, the expected minimal homopolymer length based on a median of seven measurements per event. Examples were included per interval of three measurements, the minimal event length, to keep the size of the database limited but still guarantee at least one example per event. The database thus contained complete homopolymer stretches, complete non-homopolymer stretches and stretches containing both non-homopolymers and homopolymers.

2.6 Network training and validation

All networks were trained for 10,000 iterations on sets of examples with an approximate homopolymer content of 40% to 45%. These networks were validated on stretches of 14,980 measurements in 11,940 reads of the validation set. A total of 315 reads were excluded from the original validation set because of a raw signal length less than 14,980 measurements. The length was set to 14,980 as validation on stretches of this length gave similar results to validation on full reads, within an acceptable time span. Stretches were randomly selected to avoid possible bias to certain positions in the raw signal. The true percentage homopolymers varied from 2.12% to 2.27% per stretch. Network performance was assessed on precision, recall, accuracy and F1 score, which is the harmonic mean between precision and recall. Networks with an accuracy of 0% were not taken into account as they produced Not-a-Number values during training, thus were not trained successfully. Networks were considered to have no predictive value if both precision and recall were 0%, or if recall was 100% and precision was equal to the percentage homopolymers in the *E. coli* genome. The two networks with either highest precision or recall were selected for further training of an additional 100,000 iterations, while monitoring performance on the training set and the validation set to avoid overfitting. Finally, the best performing model was tested on randomly selected stretches of 14,980 measurements from the test set. A total of 11,800 stretches were selected as 456 of the 12,256 reads did not have the minimal required length of 14,980.

The best performing network was incorporated in the tool catfish based on the following selection criteria: (i) a recall of approximately 75% and (ii) a precision of at least 10%. The optimal decision threshold for the best network was based on the ability of accurately detecting homopolymers instead of single homopolymer measurements (**Supplementary Figure 3**). Basecallers and polishers require input to have a certain size. It is therefore not essential to call every measurement to select the homopolymer. Actual performance of the selected model on detection of homopolymer regions instead of single measurements was assessed on one hundred full reads with a combined length of 33,537,616 measurements. The underlying base sequence of predicted homopolymers was inspected on position, length in measurements, length in bases and base composition.

2.7 Tool testing

Finally, the tool was tested on a set of 1,000 reads to compute CPU time needed per read for homopolymer detection and signal splitting. The average CPU time was computed after four replicates of testing. This showed what the most expensive step in the process was. The tool was tested on another 100 reads to estimate total CPU time and disk space needed.

Additionally, different Albacore settings were tried to find to optimal settings for homopolymer basecalling. A set of 1,000 reads was basecalled with Albacore using different settings for maximum chunk size and homopolymer correction. It was reported that homopolymer calling by Albacore could benefit from an increased chunk size over the default 1,000, as the chunks will be able to span over low complexity regions. Homopolymer correction was deactivated by default. Four replicates were performed. CPU times were measured and basecaller performance was assessed on the number of insertions, deletions, mismatches and matches.

3 Results

Nanopore sequencing suffers from a relatively high error rate compared to SGS technologies. Especially the identification of homopolymers has proven to be a difficult task. Two different neural network architectures, the RNN and the ResNetRNN, were employed with the goal to accurately detect homopolymers in the raw signal generated by a MinION sequencer. Hyperparameter settings were selected by random search. Performance was evaluated and the best performing network was implemented in a tool for homopolymer selection.

3.1 ResNetRNNs have a greater capacity in homopolymer detection than RNNs

A total of 67 networks, having either the RNN or ResNetRNN architecture, were trained and compared on performance to determine which architecture was most adequate in homopolymer detection (**Supplementary Table 4**). Two RNNs were omitted because they were not successfully trained. In general, ResNetRNNs outperformed RNNs based on precision and recall, although the difference was limited (**Figure 2**). RNNs reached a maximum recall of 76.92% and precision of 12.69%, while ResNetRNNs reached a maximum recall of 78.47% and precision of 14.58%. The relative high recall and low precision of most networks was indicative of many falsely predicted homopolymer measurements. Networks with the highest accuracies (close to 98%) had an extremely poor precision and recall due to the fact that these networks consistently predicted non-homopolymers in all cases. As the true percentage homopolymers in the *E. coli* genome is about 2.2%, accuracy is high when predicting exclusively homopolymers, but these networks do not have any useful predictive value. Additionally, the best performing ResNetRNN estimated a homopolymer content of 10.9% against a 13.4% predicted content by the best performing RNN. Thus, the hybrid ResNetRNN architecture, given the proper hyperparameters (see **3.2 The right combination of hyperparameters is essential for performance**), was better able of discerning homopolymer measurements from non-homopolymer measurements than the RNN architecture.

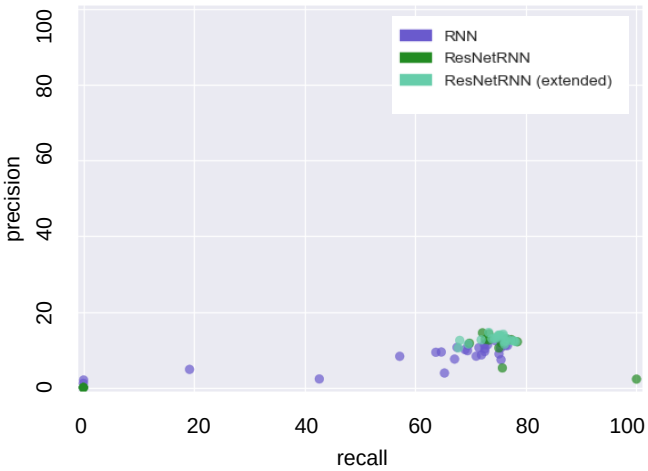

Figure 2 - Network performance. Performance of networks with either a RNN architecture or a ResNetRNN architecture based on precision and recall.

3.1.1 Extending well-performing RNNs with a ResNet improves the performance

Besides the common method of random search for hyperparameter optimisation, another approach was taken as well: the extension of a ResNetRNN. It combined randomly selected ResNet hyperparameters with predefined hyperparameter settings for the RNN substructure of the ResNetRNN based on well-performing RNNs. Following this method, five new instances of a ResNetRNN for each of the top four RNNs were trained. Thirteen of the twenty ResNetRNNs had an improved performance, measured in F1 score, in comparison to the original RNNs, of which ten performed better than the best RNN (**Supplementary Figure 4**). All hybrid networks except one had an increased performance with regard to the RNNs not used for extension, although no extreme increase or decrease in performance was measured. Still, this extension approach was beneficial in the problem of homopolymer detection as the majority of extended ResNetRNNs achieved better results than the original RNNs.

3.1.2 Performance of the networks is variable

Performance of all the networks varied considerably depending on the hyperparameter combinations (**Figure 3**). Similar top results were reached in both approach for ResNetRNN hyperparameter optimisation as the best performing networks both had a precision of approximately 15%

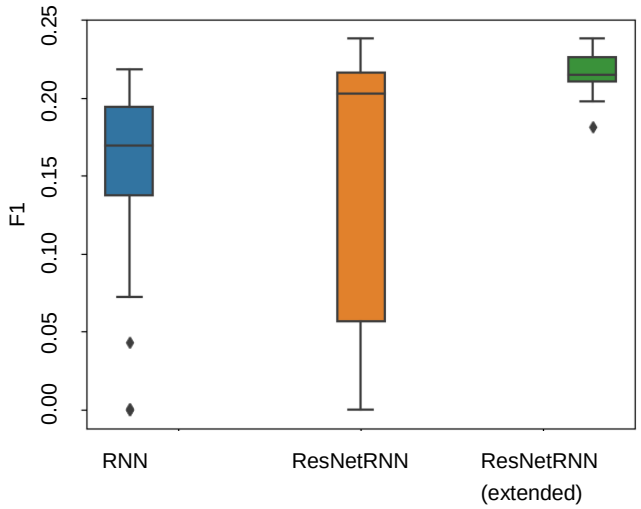

Figure 3 - Boxplot of F1 score for network architecture. Random search was applied for hyperparameter optimisation for RNNs (blue) and ResNetRNNs (orange). Additionally, the approach to extend RNNs to ResNetRNNs (green) was applied.

and recall of 73%. However, networks created by the extension method showed more consistent performance than the random RNNs and ResNetRNNs. This was expected, because of the shared basis of the extended ResNetRNNs. Especially the randomly generated ResNetRNNs showed varied performance. A quarter of the random ResNetRNNs did not have predictive value, while others were amongst the best performing. RNNs generally achieved a lower performance, which was caused by both a lower precision and recall. It was clear that a good performance does not simply depend on the network architecture, but was heavily influenced by the hyperparameters settings.

3.1.3 Additional training improved performance by a small margin

Two networks, further referred to as ResNetRNN 1 and 2, were selected for additional training of 100,000 iterations. The learning curves on training and validation accuracy were monitored to prevent overfitting on the data. No overfitting was observed based on accuracy. The model at an earlier point in training was restored however, because training for more iterations had decreased precision (**Supplementary Figure 5**). Before training, ResNetRNN 1 had achieved the highest precision and ResNetRNN 2 the highest recall. After the additional training, both recall and precision were approximately equal for the two networks. ResNetRNN 1 was selected over ResNetRNN 2 to be incorporated into the tool catfish as it had a lower complexity than ResNetRNN2. Less computations are needed using ResNetRNN 1, which is likely to result in a faster model. ResNetRNN 1 is a 3-layer RNN with 64 units per layer, combined with 2 ResNet layer of 32 units each and a final fully connected layer (**Supplementary Figure 6)**. This model has an accuracy of 90.15%, precision of 15.21% and recall of 74.75% on homopolymer measurement detection.

The networks discussed thus far were trained on a set of examples with an overall homopolymer content of about 40%. Training the best network from start on a set less balanced resulted in higher precision (> 20%), but lower recall (<70%). Training on an even more unbalanced set improved precision and diminished recall further, Also, resuming training on a set with higher homopolymer content after training performance declined on the initial training set, resulted in a steady decrease in recall and rise in precision (**Supplementary Figure 7**).

3.2 The right combination of hyperparameters is essential for performance

The importance of the individual hyperparameters on the performance of the networks was evaluated after the initial training of 10,000 iterations for hyperparameter optimisation (**Supplementary Figure 8 and 9**). The hyperparameters optimiser, number of RNN layers, size of RNN layers, mini-batch size, number of ResNet layers, number of ResNet layers and dropout did not have a strong independent effect on performance. Learning rate seemed to have a small individual effect. A lower learning rate was associated with a better performance, especially in ResNetRNNs. A learning rate of 0.1 in combination with a larger number of layers (4 or 5) and large layer size (128 or 256 units) often resulted in networks that did not learn to detect non-homopolymers for both the RNN and ResNetRNN architecture. Alternatively, a single ResNetRNN with a learning rate of 0.1 detected solely homopolymers. This indicated that a high learning rate does not allow the networks to learn the details that discriminate a homopolymer from a non-homopolymer. Interestingly, the learning rate of 0.001 resulted in better performing networks that the lower learning rate of 0.0001, which could be due to inadequate training. The initialisation also affected the networks. For instance, two RNNs were trained that incidentally had the same hyperparameter settings (**Supplementary Table 4**). One network outperformed the other on all measures. The difference between these networks is solely due to the random parameter initialisation and training on different training examples.

RMSProp is an often-used optimiser in RNNs, while Adam is usually the preferred choice in CNNs. In the case of a combined network, the optimal choice is not apparent. On inspection of the optimiser choice, no obvious difference was present for neither RNNs nor ResNetRNNs based on optimiser alone. Although the networks that stood out because of obvious worse performance than the other networks were trained using the RMSProp optimiser, so were some of the top performing networks.

Overall, no individual hyperparameter had a major influence on network performance, although learning rate was an indicator. It is the combination of hyperparameters that is most important for network performance.

3.3 The network detects homopolymers with low precision

The purpose of the model was to detect homopolymer regions accurately. The detection of homopolymer regions was evaluated through an assessment of 100 full reads (**Figure 4**). Precision and recall for homopolymer regions instead of the measurements making up a homopolymer were calculated for different decision thresholds. In all cases, the false discovery rate was extremely high. A threshold of 0.9 was therefore selected to restrain the number of falsely detected homopolymers to some extent. At this setting, about two thirds of all true homopolymers are detected. The number of homopolymers was overestimated with a false discovery rate of 67.7%. About 5% of the falsely identified homopolymers was a stretch of four identical bases, about 15% of three identical bases and about 25% of two identical bases. Also, the prevalence of small

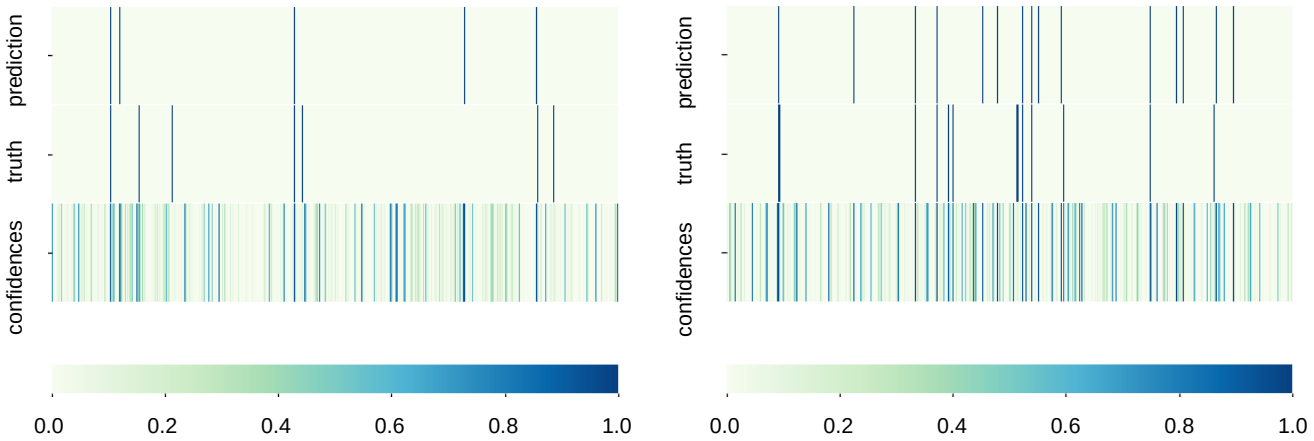

Figure 4 - Comparison of homopolymer predictions against genuine homopolymer positions in two full reads. Decision threshold is set at 0.9. Top row: predicted positions of homopolymers; middle row: actual homopolymer positions; bottom row: confidences of predictions as indicated by the colour scale.

repetitive sequences like 'ACACAC' was observed, but repetitive sequences were only an extremely small fraction of incorrectly predicted homopolymers. About half of the missed homopolymers had a predicted confidence score over 0.6, which indicates that the network does get some indication of a homopolymer being present. Predicted homopolymer regions had a shorter length than in reality. Homopolymer regions were underestimated on both sides for ten to fifteen measurements and usually overlapped with less than five bases. Thus, most predicted homopolymer regions only partially represented the true homopolymers. On inspection of the overall base composition of the predicted homopolymer regions, there was no indication of bias for detecting homopolymer of certain base compositions (**Supplementary Figure 10**). Homopolymers composed of adenine and thymine are naturally more occurring that cytosine or guanine homopolymer. Although the low precision of the model was not desired, the model was used as selector of homopolymers to estimate the possible gain by use of catfish.

3.4 A specialised homopolymer caller is needed

The command line tool catfish was designed as a selection tool that finds homopolymer regions in the raw nanopore signal and splits the signal in chunks of minimal 100 to a user-defined maximum. As catfish centers the homopolymers in the split, this has the advantage that the complete homopolymer is contained even though the model underestimates the homopolymers in length. At the moment, it takes the tool approximately 31 seconds per read in CPU time for homopolymer analysis and an additional 8 seconds in CPU time for splitting. When splitting the reads, a sizeable proportion of the metadata is copied as well, resulting in a steep increase in needed disk space. A set of 100 reads was split into 19,274 reads that took up 425 times as much disk space. In total, catfish takes just over 40 seconds in CPU time per read. In the ideal situation in which reads are solely split on true homopolymers, the CPU time taken for splitting of the reads is halved.

The purpose of this pre-processing step is to be able to basecall the homopolymer containing stretches with a specialised basecaller to increase accuracy, while saving time and computational expense on non-homopolymer stretches by basecalling them regularly. We looked into Albacore 2.3.3 and explored different settings to improve homopolymer calling. A comparison on using different chunk sizes with homopolymer correction activated, showed that increasing the chunk size decreased the number of deletions with respect to the default (**Figure 5**). The effect was the strongest when increasing chunk size to 20,000. Larger chunk sizes did not significantly decrease the number of deletions further. Interestingly, the number of insertions increases when increasing the chunk size to 10,000, but further increasing the chunk size showed slightly less insertions. A similar pattern was observed for the number of mismatches and matches. The proportion of deletions, insertions, matches and mismatches however stayed constant for every chunk size and activated or deactivated homopolymer correction, and the total number of called bases decreased with increased chunk size. CPU times steadily increased for every increase of 10,000 for chunk size. For homopolymer detection, Albacore can be optimised a little by activating homopolymer detection and increasing chunk size to 10,000. At a chunk size of 20,000, the total number of bases had decreased by 10%. At the settings of chunk size at 10,000 and homopolymer correction activate, the number of deletions decreased with 3% within a reasonable CPU time. However, the number of insertions increased with the same proportion as the number of deletions decreased. The effect of the adjusted setting was thus limited. A basecaller specialised in homopolymer calling must be developed to exploit the advantage of pre-processing the raw signal on homopolymers.

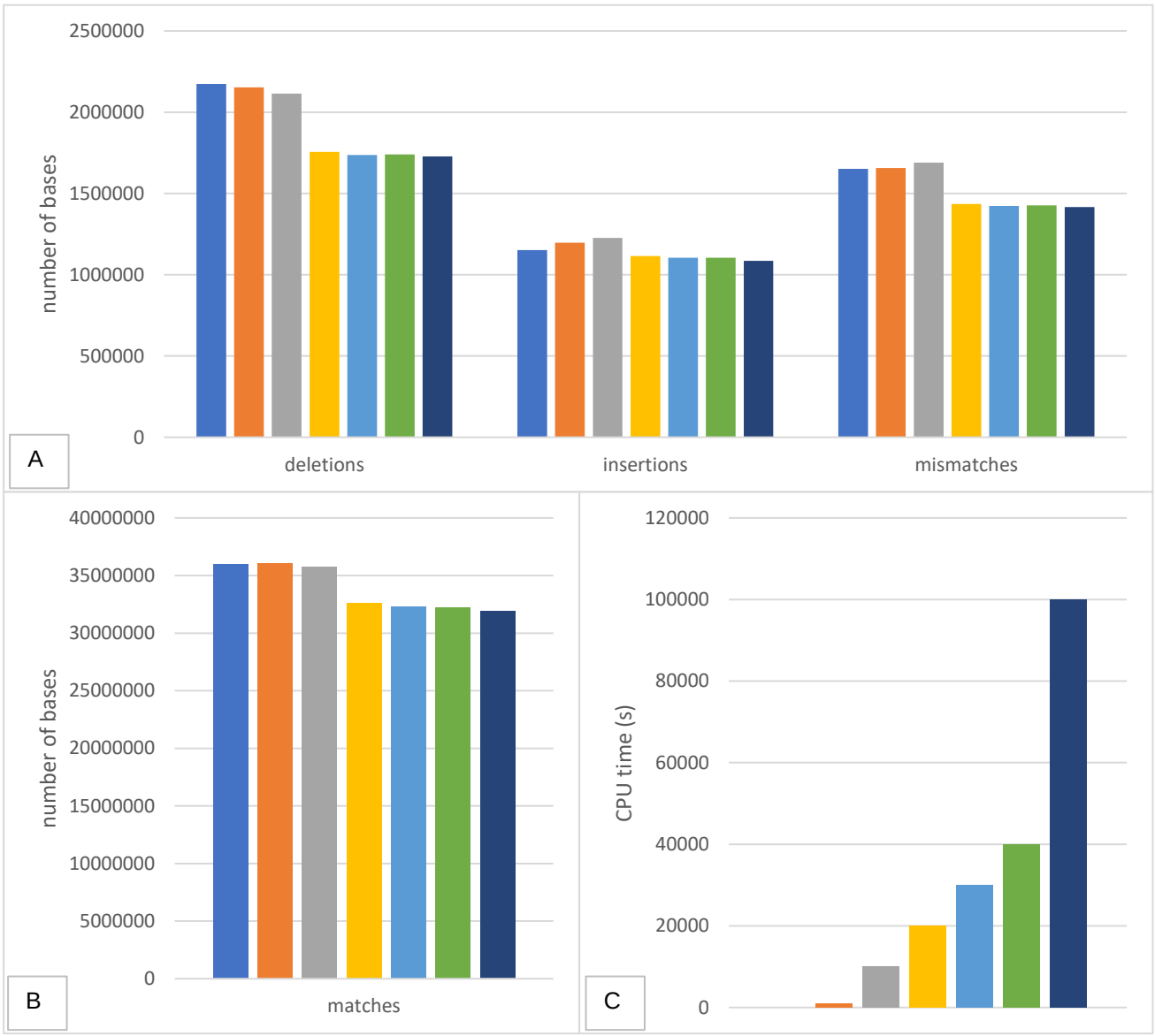

Figure 5 – Comparison of Albacore on different settings. Basecalling performance on number of deletions, insertions, mismatches (A), matches (B) and CPU time (C) using default settings (blue) versus homopolymer correction in combination with a chunk size of 1,000 (orange), 10,000 (grey), 20,000 (yellow), 30,000 (light blue), 40,000 (green) or 100,000 (dark blue) was measured. CPU time for default settings is not clearly visible in the graph as it was less than 5,000 seconds.

4 Discussion

Nanopore sequencing is a quickly developing technique for generating long reads, with the potential to advance the field. However, it suffers from a relatively high sequencing error rate compared to the current golden standard SGS techniques. The calling of homopolymers has proven to be difficult and remains the major problem in nanopore sequencing. Addressing this problem will increase the applicability of nanopore sequencing to current problems in genotyping (Goldstein *et al.*, 2019), structural variant detection (Sedlazeck *et al.*, 2018) and single nucleotide variant detection (Rang *et al.*, 2018).

In this research, two different types of neural networks were trained on a MinION generated *E. coli data* set with the intent to accurately detect homopolymers based on the raw signal. A comparison on the different neural networks showed variable performance for the two architectures, which emphasized the role of the combination of hyperparameters. Individual hyperparameters did not have a major effect by themselves. In general, ResNetRNNs had a greater capacity to detect homopolymers that the RNNs. Although the differences in accuracy, precision and recall were not large between the different network architectures, applying a more complex network with deeper learning capabilities led to more accurate predictions. A ResNetRNN was further developed by additional training and moving the decision threshold. It achieved a reasonable recall but

fairly low precision, as the model overestimated the number of homopolymers present in the raw signal considerably. In light of the low percentage of homopolymers in average reads, it was expected that the number of homopolymers would be overestimated. The networks were trained on a limited number of examples. In particular, most non-homopolymer examples were not included in training as the training set was almost balanced. By chance, instances that contained important differences between homopolymers and non-homopolymers may have been excluded or not been included in the training set. However, with respect to the available examples in the database, the networks were likely to have learned all possible information as considerable additional training did not improve performance.

As a practical application, the neural network was incorporated in the pre-processing tool catfish that splits the raw signal on homopolymer stretches, so they can be processed with extra care to improve the calling of homopolymers, while saving computational power and time by regularly processing non-homopolymer stretches. Detecting homopolymers in the nanopore reads is a relatively costly operation and the tool would twofold increase the basecalling process in CPU time. At the moment, there is no sufficient basecaller or polisher that can handle homopolymer calling. Albacore 2.3.3 and later do have a homopolymer correction settings, although the effect appeared to be small. Recently, ONT released the latest research basecaller Flappie, which uses a new algorithm that supposedly handles homopolymers better (Brown, 2018). An initial comparison revealed the basecaller to more accurately call homopolymers up to six bases in length, but no significant improvement for longer homopolymers (Robison, 2018). It is clear that there is still much room for improvement in basecalling software. In the future, we see homopolymer selection joined with specific homopolymer calling, incorporated in standard basecallers. This would additionally solve the issue with the large disk space required, because the splitting step would be omitted.

The first step to advance catfish is optimising the network as the current network lacked the desired precision. When constructing a neural network, choosing the optimal hyperparameters is crucial to generate well-performing models. Random search was applied for hyperparameter optimisation. Although most hyperparameters independently did not have a strong influence on network performance, the combination of hyperparameters does. This was clear from the varied performance between networks of the same and different architecture. Larger networks may have need more training to reach the same level of performance as smaller networks because they have more parameters to fine tune. Networks with a high learning rate were generally unable to detect homopolymers well, which could be expected as they are more prone to stepping over the optimal solution due to the larger step size with which the updates are made.

A small fraction of the large number of possible hyperparameter combinations was explored using random search. Bergstra & Bengio (2012) showed that neural networks on average achieved better accuracies when random search was used than a grid search of 100 trials in the optimisation of nine hyperparameters in experiments with eight or more networks on binary classification. This indicates that a sufficient number of networks is tried, although the addressed problem is different from homopolymer detection. It cannot be guaranteed that the optimal hyperparameter combination was found, even when assuming that the determined range of hyperparameter setting includes the optimal setting. A promising alternative approach is evolutionary algorithms, such as population-based training as it exploits well-performing models to explore a new range of hyperparameters dynamically during training (Jaderberg *et al*., 2017; Oehmcke & Kramer, 2018).

The alternative approach of extending a relatively well-performing RNN with a ResNet was not unequivocally better than a random search for hyperparameter optimisation. This method did found the best combination of hyperparameters. Also, none of the extended RNNs had a strong decrease in performance, which is likely due to the structure of the ResNet layers. In the worst case scenario, these layers do not learn but pass on the input without modification.

In contrast to the hyperparameters of a network that have to be selected beforehand, the parameters are learned via gradient descent during training. Appropriate examples for learning are therefore crucial. The raw signal contains measurements that capture the transition from non-homopolymers to a homopolymer. This is not reflected in the applied labelling method that strictly discriminates between homopolymer and non-homopolymer. A different labelling system using more labels could have been applied to make this distinction. For instance, the sequence ‘TAAAA’ could be the beginning of a homopolymer but can also be followed by another base than an adenine. The difference between a possible homopolymer and a true homopolymer could be exploited to increase model performance, if the transitions from non-homopolymers to possible homopolymers were marked as such. The network could possibly recognise short stretches of identical bases and assign different confidentialities as the number of identical bases increases. If a certain threshold would be reached in the sequence, it will be marked as homopolymer and otherwise it will not. This could possibly reduce the number of short stretches of identical bases identified as homopolymer.

The majority of homopolymers was detected by the model, although one third is missed and predicted homopolymers are shorter in length than the true homopolymers. It is likely beneficial to use a larger example length (was

set to 35) in future research so the network will learn to recognise longer homopolymers. Additionally, over 65% of the predicted homopolymers did not overlap with true homopolymers at all. A substantial number of these incorrectly predicted homopolymers had an underlying base sequence of two, three or four identical bases. The detection of theses short stretches of identical bases could be explained by the fact that many examples used in training represented edge cases that contained both homopolymers and non-homopolymers.

The ultimate goal is to be able to generate confident genome assemblies for all organisms. In this research, a prokaryotic set was used as opposed to a eukaryotic set to avoid complications due to DNA modifications, as modified bases give a different signal than their canonical versions (Stoiber *et al*., 2016). However, Chiron, a basecaller with a similar neural network at its core as the one integrated in the tool, was able to generalize well over a human set while trained on a small prokaryotic set (Teng *et al*., 2018). This suggests that the proposed neural network could be able of generalizing to other organisms as well.

A big obstacle in training of the networks is the severe class imbalance. Undersampling of the majority group combined with oversampling of the minority group was used to enable training on a balanced set. Other techniques to work with imbalanced sets exist as well. In some data sets over 10,000 examples in size, cost-sensitive learning has proven to be more efficient than sampling (Weiss *et al*., 2007) and easy to implement, although learning could be more difficult in heavily imbalanced problems like this one (Zhou & Liu, 2006). Alternately, a different perspective could be taken on homopolymer detection. Novelty detectors exceeded binary classifiers like the used neural networks in data for which the minority class is than 5% (Lee & Cho, 2006).

Neural networks are powerful machine learning algorithms that handle complex data. Many different architectures exist, which provide different ways of learning the expected output from a presentation of input samples. Two well-known architectures, the RNN and the ResNet, were tried, which are known for learning from sequential data and extracting local features respectively. It would be interesting to explore the capabilities of less used networks, such as the capsule network (CapsNet) (Sabour *et al*., 2017) that uses nested sets of CNN layers to extract local features.

Future directions are focused on improving the performance of the network by strengthening precision and recall. As mentioned above, different approaches exist and could be combined, including an alternative labelling method, cost-sensitive learning, different network architectures, training examples that are composed of longer stretches of signal or a dynamic hyperparameter optimisation algorithm. Nonetheless, it is difficult to pinpoint what aspects matter most with regard to network performance after the many subsequent linear and non-linear operations that transpire in a neural network. This is not the first attempt to utilise the raw signal instead of the previously used segmented signal. However, it is the first approach that solely focusses on identification of homopolymers.

In conclusion, neural networks are able to detect a pattern that discerns homopolymers from non-homopolymers in the complex MinION generated data. The ResNetRNN architecture enabled increased learning over the RNN architecture, as they are deeper networks that combine local feature learning with extraction of long-range information. Although the neural networks presented here lacked the precision required for accurate selection of homopolymer regions, we propose several venues towards further improvements in this respect. Optimising the neural networks in combination with a polisher or a basecaller specialised in calling homopolymers, has the potential to reduce the error rate of nanopore sequencing, while limiting additional costs by restricting additional care to only those regions that actually contain homopolymers.

Acknowledgements

I would like to thank Carlos de Lannoy and Dick de Ridder for their supervision and feedback.

References

Abadi, M. *et al*. (2016). TensorFlow: A system for large-scale machine learning. In *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation* (pp. 265–283). Savannah, United States: USENIX Association Berkeley.

Agah, S. *et al*. (2016). DNA sequencing by nanopores: Advances and challenges. *Journal of Physics D: Applied Physics*, *49*(41), 413001–413015.

Bergstra, J. *et al*. (2012). Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, *13*, 281–305.

Blattner, F. *et al*. (2014). *Escherichia coli* str. K12 substr. MG1655, complete genome, U00096.3. Retrieved October 3, 2018, from https://www.ncbi.nlm.nih.gov/nuccore/U00096.3

Boža, V. *et al*. (2017). DeepNano: Deep recurrent neural networks for base calling in MinION nanopore reads. *PLoS One*, *12*(6), e0178751.

Brown, C. G. (2018). Clive G Brown: Nanopore Community Meeting 2018 talk. Retrieved January 21, 2019, from https://nanoporetech.com/about-us/news/clive-g-brown-nanopore-community-meeting-2018-talk

Buda, M. *et al*. (2018). A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, *106*, 249–259.

Cho, K. *et al*. (2014). On the properties of neural machine translation: encoder-decoder approaches. In *Eighth*

Workshop on Syntax, Semantics and Structure in Statistical Translation (pp. 103–111). Doha, Qatar: Association for Computational Linguistics.

Chung, J. *et al*. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. In *Neural Information Processing Systems 2014 Deep Learning and Representation Learning Workshop* (pp. 1–9). Montreal, Canada: Neural Information Processing Systems Foundation.

Dijk van, E. L. *et al*. (2018). The third revolution in sequencing technology. *Trends in Genetics*, *34*(9), 666–681.

Elliott, I. *et al*. (2018). Oxford Nanopore MinION sequencing enables rapid whole-genome assembly of *Rickettsia typhi* in a resource-limited setting. *BioRxiv*.

Glorot, X. *et al*. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics* (pp. 249–256). Sardina, Italy: JMLR.org.

Goldstein, S. *et al*. (2019). Evaluation of strategies for the assembly of diverse bacterial genomes using MinION long-read sequencing. *BMC Genomics*, *20*(1), 23.

He, K. *et al*. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 770–778). Las Vegas, United States: IEEE Computer Society.

Heather, J. M. *et al*. (2016). The sequence of sequencers: The history of sequencing DNA. *Genomics*, *107*(1), 1–8.

Hochreiter, S. *et al*. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780.

Hochreiter, S. *et al*. (2001). Gradient flow in recurrent nets: The difficulty of learning long-term dependencies. In *A Field Guide to Dynamical Recurrent Networks*. IEEE Computer Society.

Ioffe, S. *et al*. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning* (pp. 448–456). Lille, France: JMLR.org.

Jaderberg, M. *et al*. (2017). Population based training of neural networks. *ArXiv*.

Jain, M. *et al*. (2018). Nanopore sequencing and assembly of a human genome with ultra-long reads. *Nature Biotechnology*, *36*(4), 338–345.

Kim, Y. (2014). Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing* (pp. 1746–1751). Doha, Qatar: Association for Computational Linguistics.

LeCun, Y. *et al*. (1999). Object recognition with gradient-based learning. In *Shape, contour and grouping in computer vision* (pp. 319–345). London, United Kingdom: Springer-Verlag.

Lee, H. *et al*. (2006). The novelty detection approach for different degrees of class imbalance. In Springer-Verlag (Ed.), *Proceedings of the Thirteenth International Conference on Neural Information Processing* (pp. 21–30). Hong Kong, China: Springer.

Loman, N. (2017). Thar she blows! Ultra long read method for nanopore sequencing. Retrieved September 20, 2018, from http://lab.loman.net/2017/03/09/ultrareads-for-nanopore/

Lu, H. *et al*. (2016). Oxford Nanopore MinION sequencing and genome assembly. *Genomics, Proteomics & Bioinformatics*, *14*(5), 265–279.

Misiunas, K. *et al*. (2018). QuipuNet: Convolutional neural network for single-molecule nanopore sensing. *Nano Letters*, *18*(6), 4040–4045.

Oehmcke, S. *et al*. (2018). Knowledge sharing for population based neural network training. In *Joint German/Austrian Conference on Artificial Intelligence* (pp. 258–269). Berlin, Germany: Springer, Cham.

Oxford Nanopore Technologies. (2017). New basecaller now performs 'raw basecalling', for improved sequencing accuracy. Retrieved September 21, 2018, from https://nanoporetech.com/about-us/news/new-basecaller-now-performs-raw-basecalling-improved-sequencing-accuracy

Oxford Nanopore Technologies. (2018). MinION. Retrieved September 21, 2018, from https://nanoporetech.com/products/minion

Payne, A. *et al*. (2018). BulkVis: A graphical viewer for Oxford nanopore bulk FAST5 files. *Bioinformatics*, 1–6.

Quick, J. *et al*. (2016). Real-time, portable genome sequencing for Ebola surveillance. *Nature*, *530*(7589), 228–232.

Rang, F. J. *et al*. (2018). From squiggle to basepair: Computational approaches for improving nanopore sequencing read accuracy. *Genome Biology*, *19*(1), 1–11.

Robison, K. (2018). Flappie vs. Albacore via Counterr. Retrieved January 18, 2019, from http://omicsomics.blogspot.com/2018/12/flappie-vs-albacore-via-counterr.html

Rueda, F. M. *et al*. (2018). Convolutional neural networks for human activity recognition using body-worn sensors. *Informatics*, *5*(2), 26.

Sabour, S. *et al*. (2017). Dynamic routing between capsules. *Advances in Neural Information Processing Systems*, 3856–3866.

Sárközy, P. *et al*. (2018). Calling homopolymer stretches from raw nanopore reads by analyzing k-mer dwell times. In *International Foundation for Medical and Biological Engineering Proceedings* (Vol. 65, pp. 241–244). Prague, Czech Republic: International Foundation for Medical and Biological Engineering.

Schuster, M. *et al*. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Sedlazeck, F. J. *et al*. (2018). Accurate detection of complex structural variations using single-molecule sequencing.

Nature Methods, *15*(6), 461–468.

Simpson, J. T. *et al*. (2017). Detecting DNA cytosine methylation using nanopore sequencing. *Nature Methods*, *14*(4), 407–410.

Srivastava, N. *et al*. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958.

Stoiber, M. H. *et al*. (2016). De novo identification of DNA modifications enabled by genome-guided nanopore signal processing. *BioRxiv*, 094672.

Teng, H. *et al*. (2018). Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning. *GigaScience*, *7*, 1–9.

Weiss, G. M. *et al*. (2007). Cost-sensitive learning vs. sampling: Which is best for handling unbalanced classes with unequal error costs? In *Proceedings of the 2007 International Conference on Data Mining* (Vol. 7, pp. 35–41). Las Vegas, United States: Springer.

Wick, R. R. *et al*. (2018a). Deepbinner: Demultiplexing barcoded Oxford Nanopore reads with deep convolutional neural networks. *PLoS Computational Biology*, *14*(11), e1006583.

Wick, R. R. *et al*. (2018b, March 5). Comparison of Oxford Nanopore basecalling tools. Retrieved January 18, 2019, from https://github.com/rrwick/Basecalling-comparison/

Zhou, Z.-H. *et al*. (2006). Training cost-sensitive neural networks with methods addressing the class imbalance problem. *IEEE Transactions on Knowledge and Data Engineering*, *1*, 63–77.

Supplementary Information

Supplementary Figures

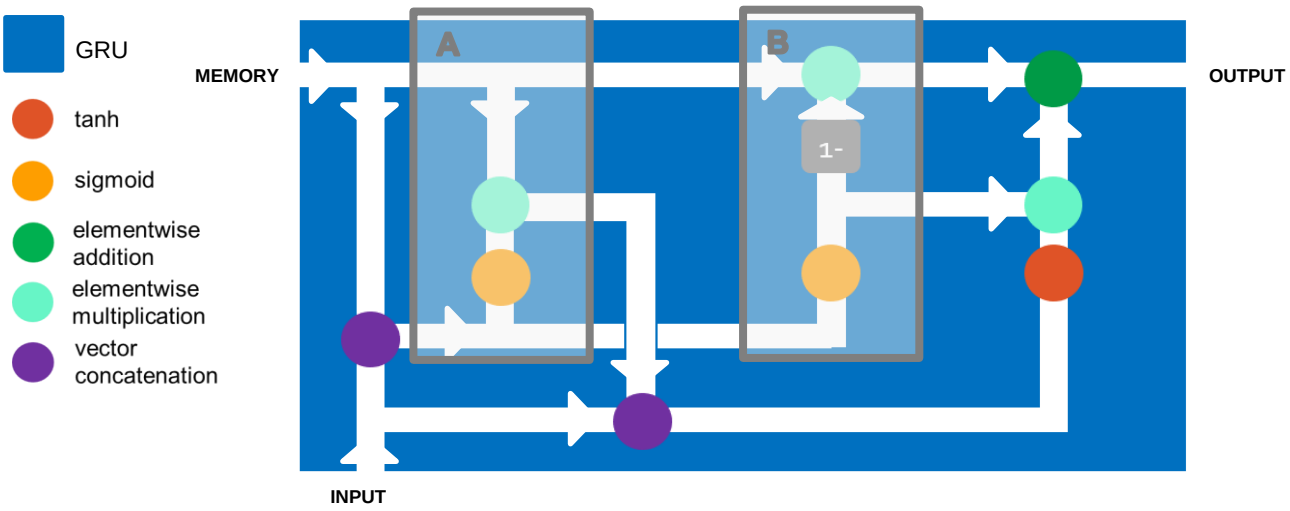

FIGURE 1 - GRU. A GRU MAKES USE OF TWO GATING MECHANISMS: THE RESET GATE (A) AND THE UPDATE GATE (B). INPUT AND PREVIOUS MEMORY GO THROUGH THE RESET GATE. THE RESET GATE DECIDES WHAT INFORMATION FROM THE PAST TO FORGET. CANDIDATE MEMORY UNDERGOES TANH ACTIVATION. MEMORY IS UPDATED GOING BY THE UPDATE GATE. IT DECIDES WHAT INFORMATION FROM THE PAST TO ADD TO KEEP IN THE FUTURE. THE UPDATED MEMORY AND CANDIDATE MEMORY ARE ADDED TO FORM THE OUTPUT OF THE GRU, WHICH IS ALSO THE NEW MEMORY OF THE UNIT.

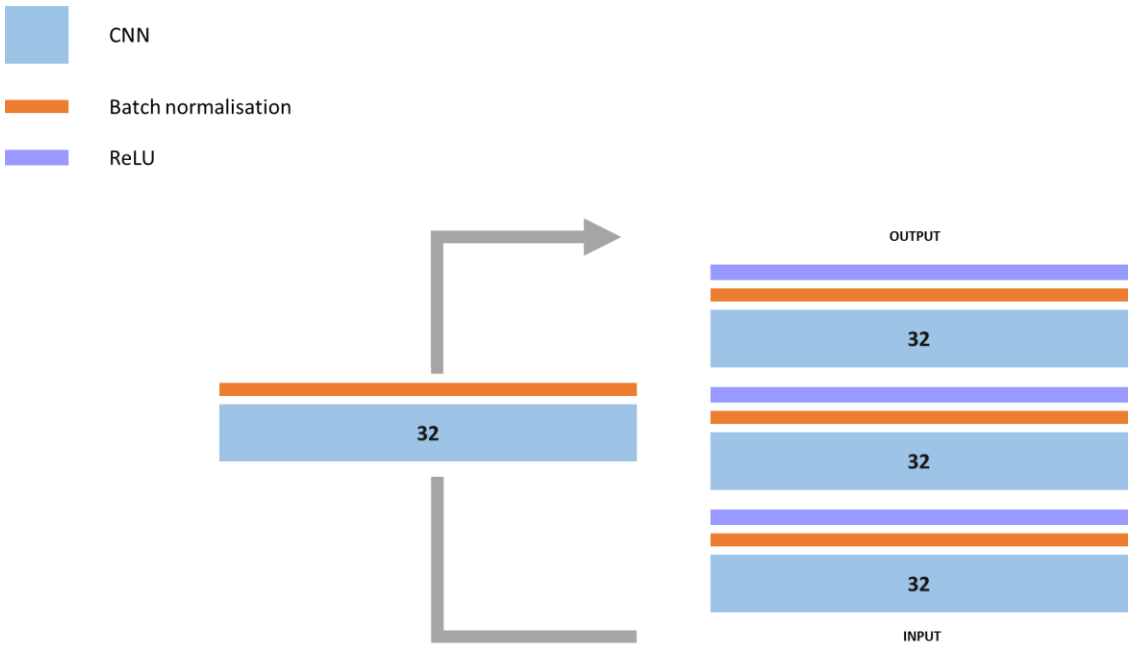

FIGURE 2 - RESNET BLOCK. A RESNET BLOCK CONSISTS OF THREE LAYERS THAT EACH CONTAIN A CONVOLUTIONAL LAYER FOLLOWED BY BATCH NORMALISATION AND THE RELU ACTIVATION FUNCTION. THE SKIP CONNECTION IS REPRESENTED BY THE CONVOLUTIONAL LAYER FOLLOWED BY A BATCH NORMALISATION LAYER ENCLOSED IN THE ARROWS. THE SKIP CONNECTION ADDS THE IDENTITY MAPPING FROM THE INPUT TO THE TRANSFORMED OUTPUT, WHICH ALLOWS FOR BUILDING DEEPER NETWORKS.

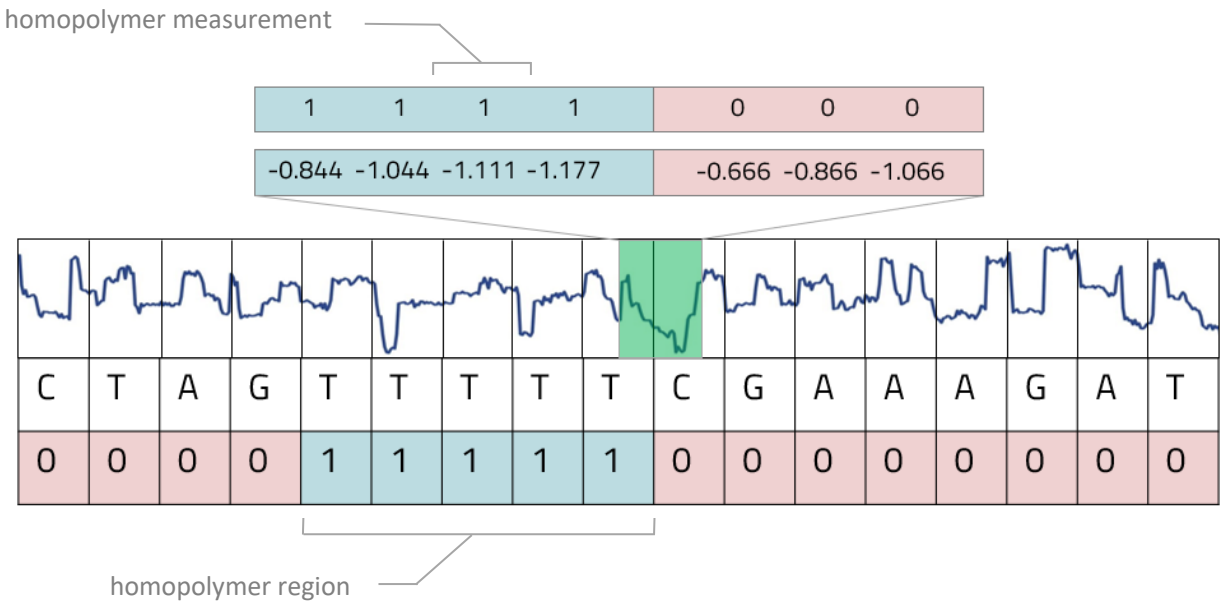

FIGURE 3 - LABELLING METHOD. A RAW SIGNAL IS DEPICTED AS A SQUIGGLE. FIRST, BASES ARE MARKED AS BEING PART OF A HOMOPOLYMER (1) OR NOT (0). SECOND, MEASUREMENTS THAT ARE LINKED TO A CERTAIN BASE ARE LABELLED ACCORDINGLY. THE GREEN BLOCK HIGHLIGHTS A PART OF THE SQUIGGLE AND SHOWS THE UNDERLYING MEASUREMENTS AND THE ASSIGNED LABELS.

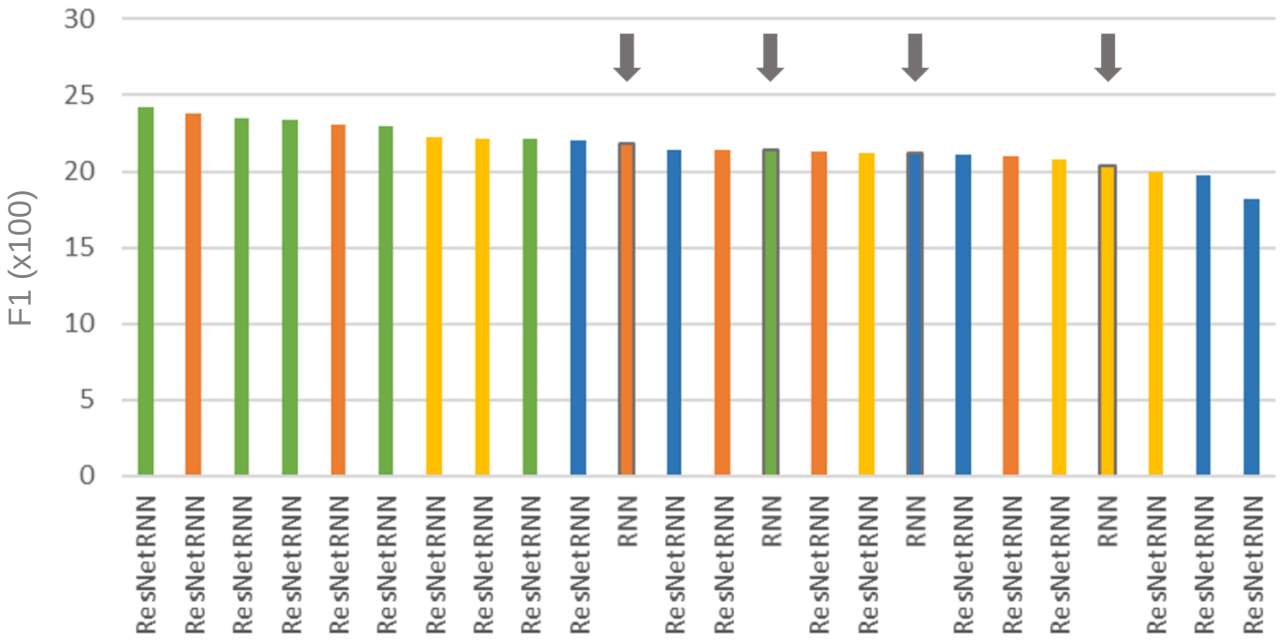

FIGURE 4 - F1 SCORE FOR RESNETRNNS BASED ON TOP PERFORMING RNNS. ARROWS INDICATE THE RNNS THAT WERE THE BASIS FOR THE RESNETRNNS. COLOURS INDICATE THE ORIGINAL RNN ON WHICH A RESNETRNN IS BASED.

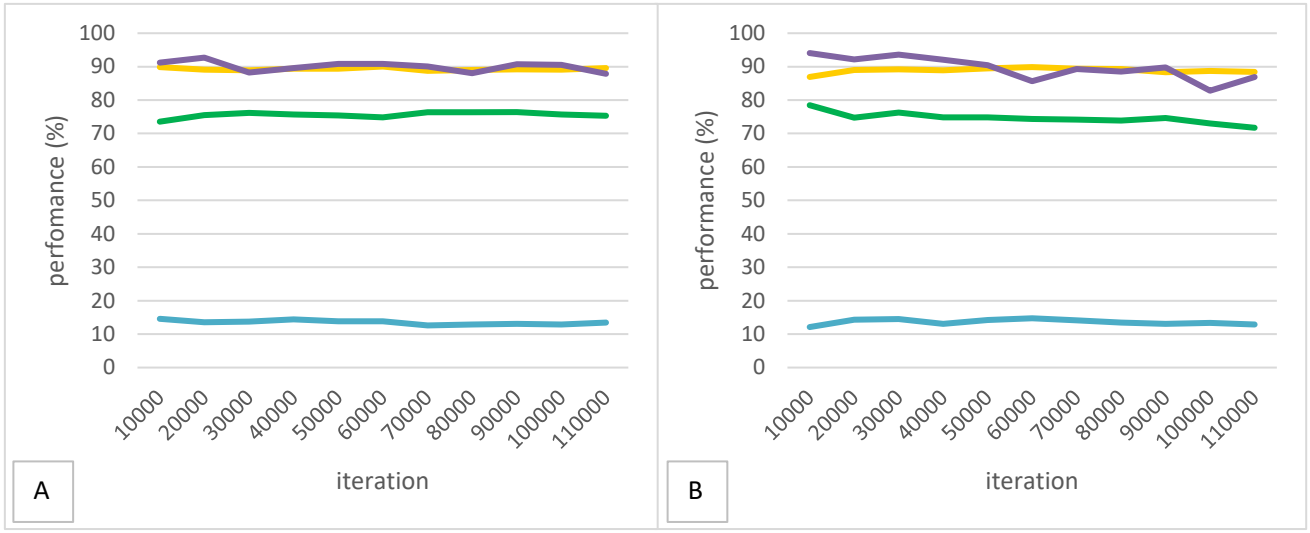

FIGURE 5 - LEARNING CURVES. THE TOP TWO RESNETRNNS WERE TRAINED FOR 110,000 ITERATIONS IN TOTAL. TRAINING ACCURACY (PURPLE), VALIDATION ACCURACY (YELLOW), RECALL (GREEN) AND PRECISION (BLUE) FOR EVERY 10,000 ITERATIONS IS DEPICTED. A) RESNETRNN 1; B) RESNETRNN 2.

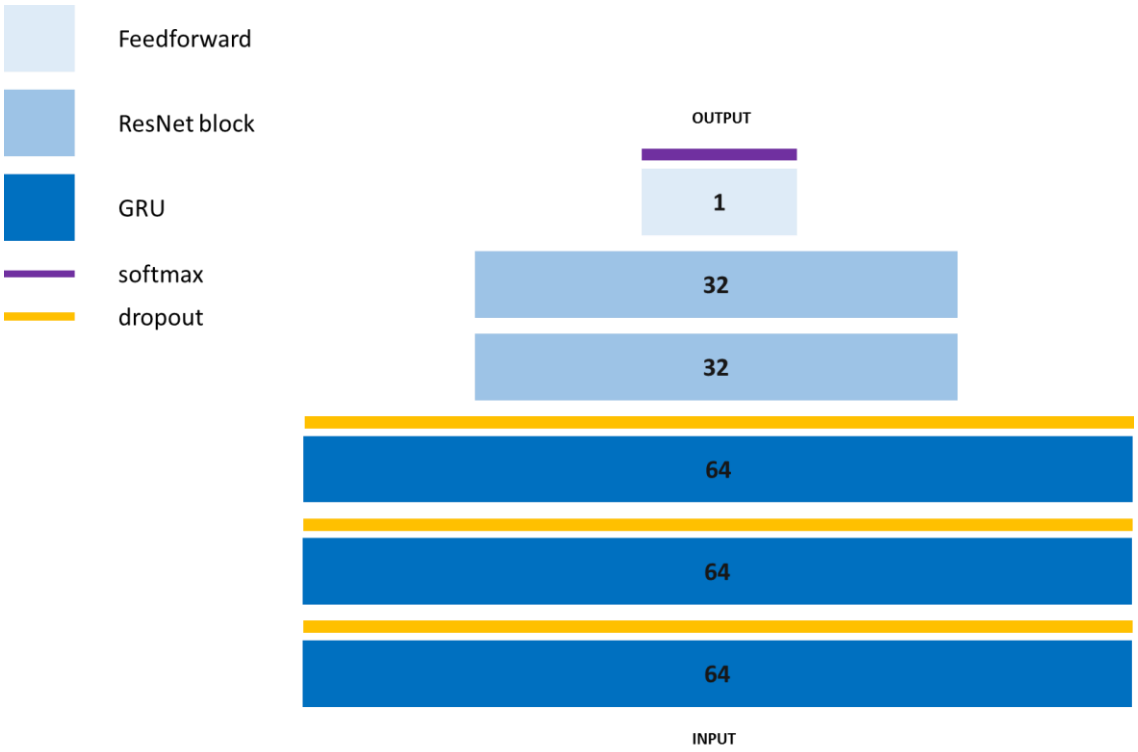

FIGURE 6 - RESNETRNN. INPUT IS TRANSFORMED BY THREE GRU LAYERS OVER WHICH DROPOUT IS APPLIED IN TRAINING. THE TRANSFORMED INPUT IS PASSED ON TO TWO RESNET BLOCKS. A FINAL FEEDFORWARD LAYER FOLLOWED BY A SOFTMAX LAYER PRODUCES THE OUTPUT.

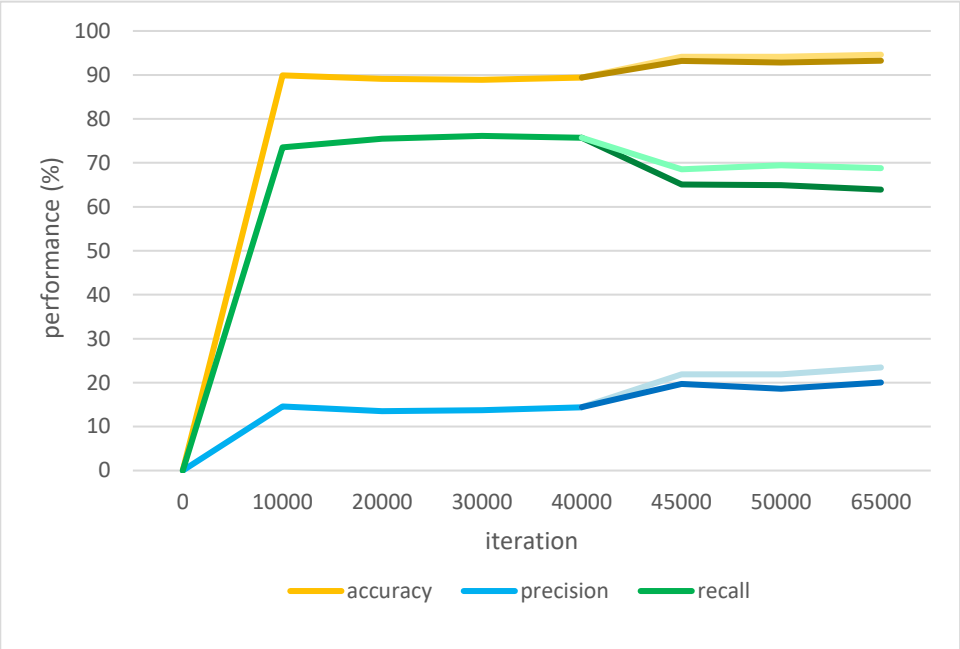

FIGURE 7 - NETWORK PERFORMANCE. THE NETWORK WAS TRAINED ON A SET WITH A HOMOPOLYMER CONTENT OF 40% FOR 40000 ITERATIONS. TRAINING WAS RESUMED WITH A SET CONTAINING 20% HOMOPOLYMERS (LIGHT) OR 30% HOMOPOLYMERS (DARK).

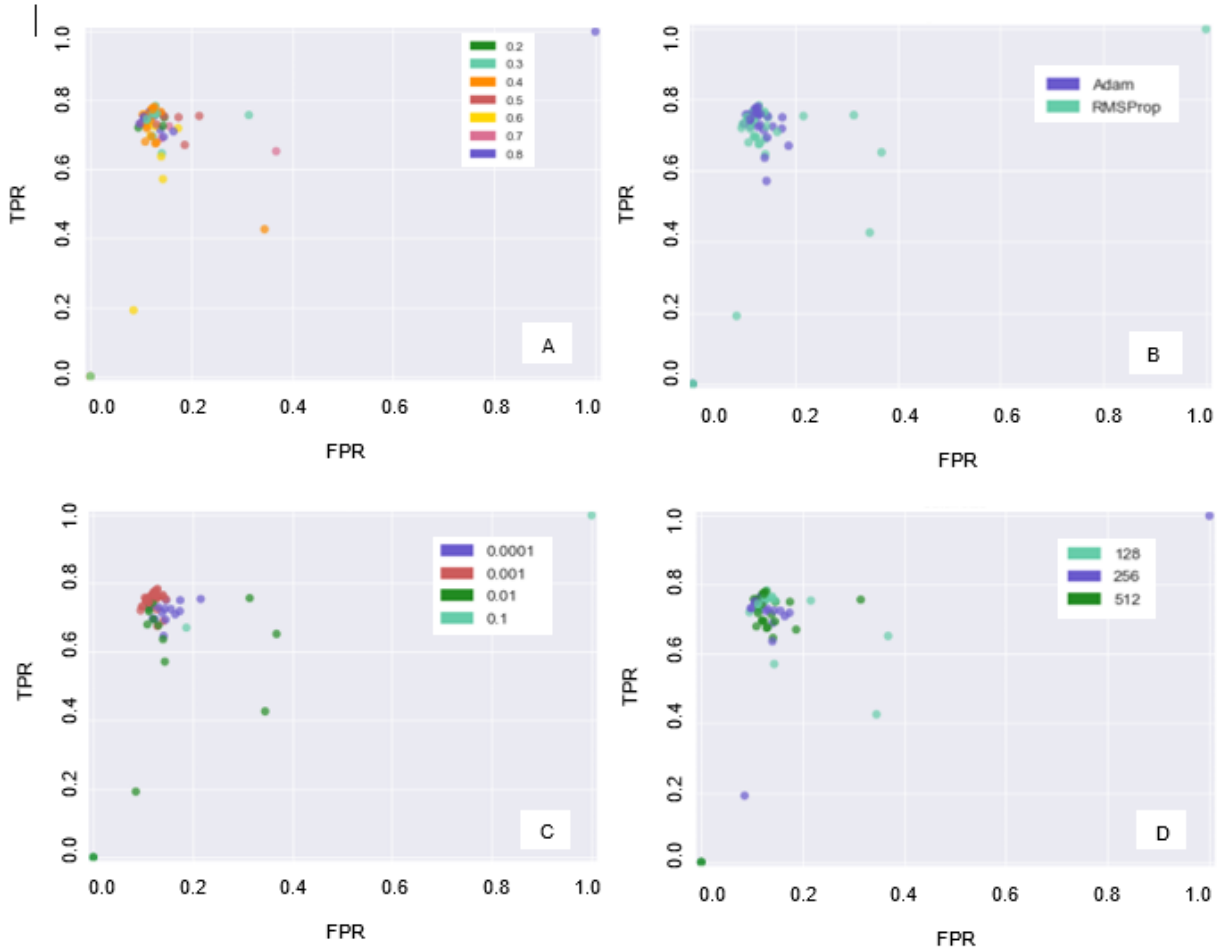

FIGURE 8 – HYPERPARAMETERS (I). NETWORK PERFORMANCE BASED ON TRUE POSITIVE RATE (TPR) AND FALSE POSITIVE RATE (FPR) COLOURED BY HYPERPARAMETER SETTINGS. A) KEEP PROBABILITY; B) OPTIMISER; C) LEARNING RATE; D) BATCH SIZE.

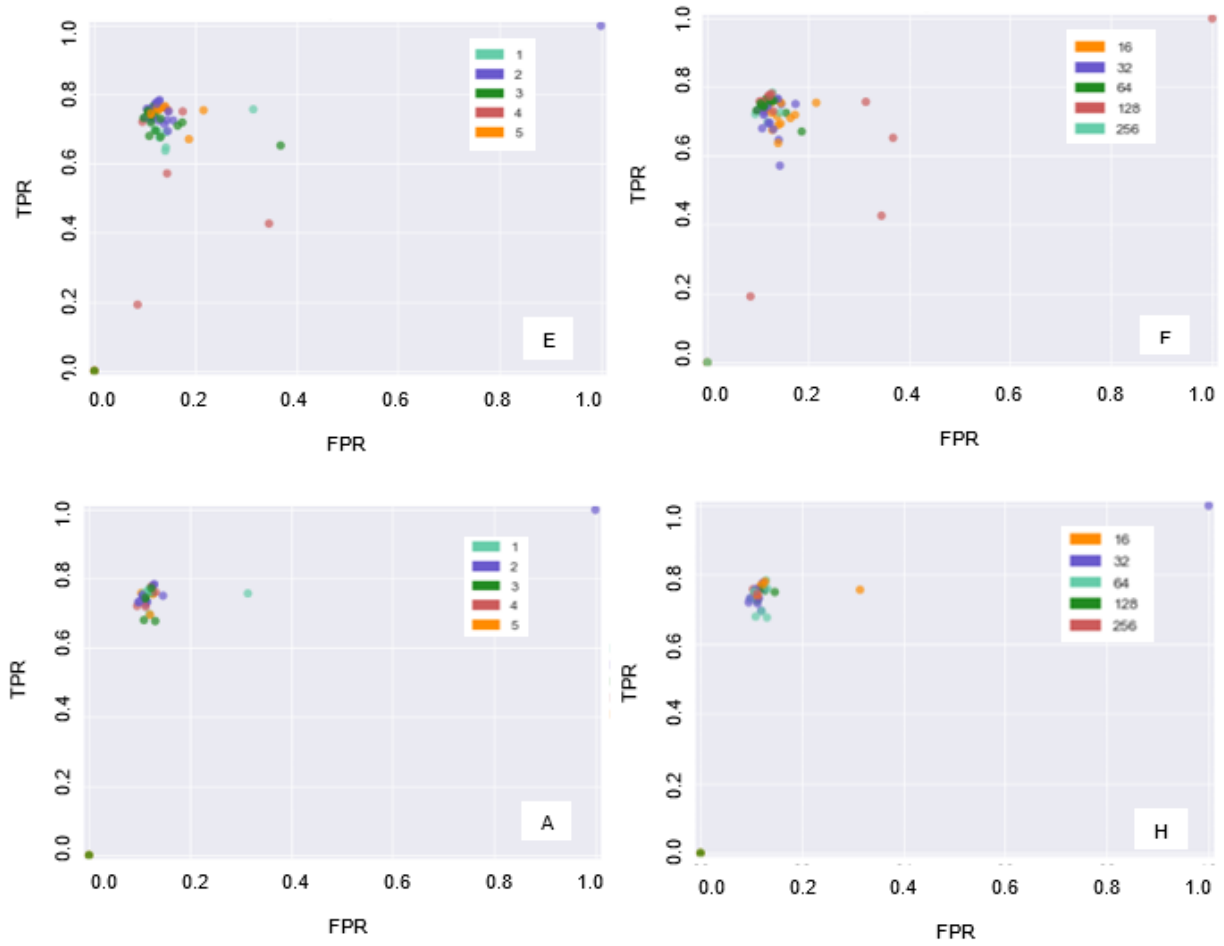

FIGURE 9 – HYPERPARAMETERS (II). NETWORK PERFORMANCE BASED ON TRUE POSITIVE RATE (TPR) AND FALSE POSITIVE RATE (FPR) COLOURED BY HYPERPARAMETER SETTING. E) NUMBER OF RNN LAYERS; F) SIZE OF RNN LAYERS; G) NUMBER OF RESNETRNN LAYERS; H) SIZE OF RESNETRNN LAYERS.

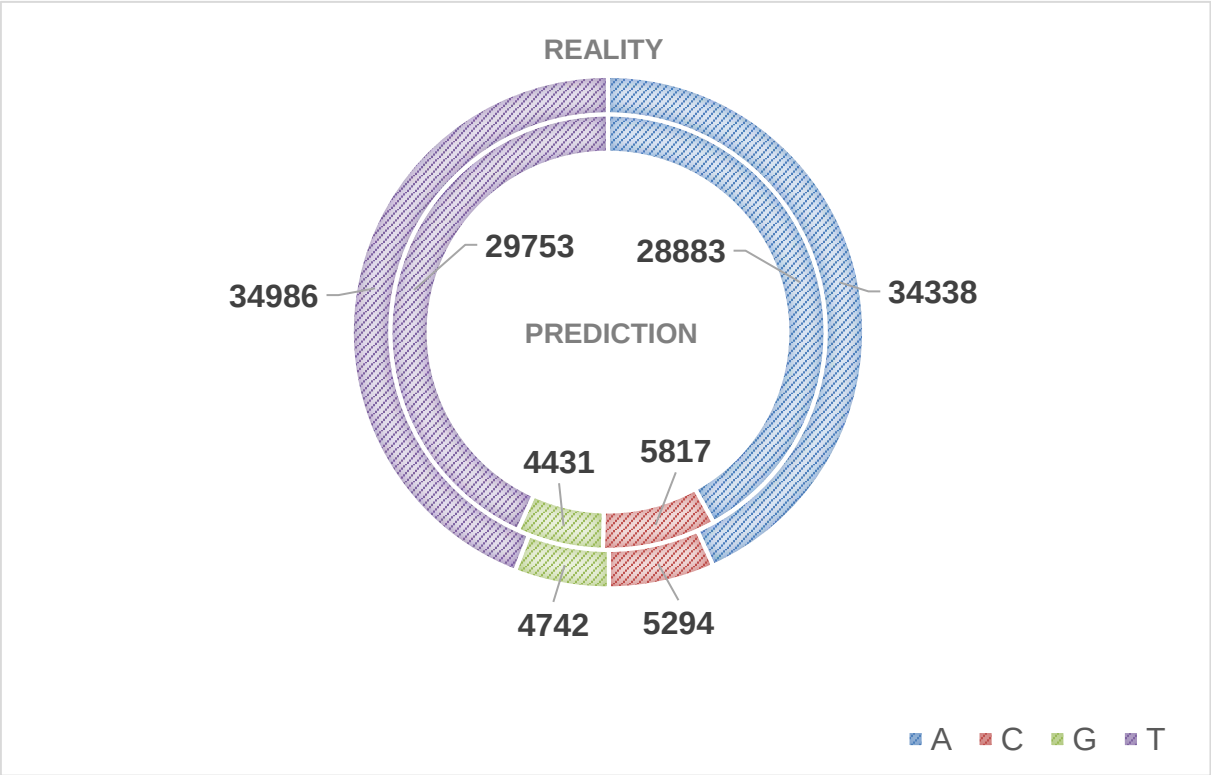

FIGURE 10 – BASE COMPOSITION OF GENUINE AND PREDICTED HOMOPOLYMERS. THE PROPORTION OF A CERTAIN BASE OVER ALL GENUINE AND PREDICTED HOMOPOLYMERS IS DEPICTED, AS WELL AS THE ABSOLUTE NUMBERS.

Supplementary Tables

TABLE 1 - DESCRIPTION OF THE HYPERPARAMETERS.

Hyperparameter	Description
Learning rate	Step size with which the model parameters are updated with respect to the loss gradient.
Optimiser	Algorithm that updates the model parameters in response the output of the loss function.
Mini-batch size	Number of training examples utilized in one iteration, after which an update of the model parameters is performed.
Layer size	The number of units that make up a single layer, which is the collection of units at a specific depth in the network.
Number of layers / blocks	The number of layers or blocks of which the RNN or ResNet respectively consists.
Dropout rate	Proportion of units temporarily removed from training. Before every mini-batch of training, units are randomly selected to drop out.

TABLE 2 - OVERVIEW OF RANGES PER HYPERPARAMETER. NO DROPOUT WAS APPLIED TO THE RESNET LAYERS.

	Options	
Network type	biGRU-RNN	ResNet
Learning rate	0.0001, 0.001, 0.01, 0.1	
Optimiser	Adam, RMSprop	
Mini-batch size	128, 256, 512	
Layer size	16, 32, 64, 128, 256	
Number of layers / blocks	1, 2, 3, 4, 5	
Dropout rate	0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8	

TABLE 3 - COMPARISON OF READ DATA BEFORE AND AFTER CORRECTION. THE TOMBO RESQUIGGLE ALGORITHM WAS APPLIED TO CORRECT THE DATA ORIGINALLY BASECALLED WITH ALBACORE 0.8.4. PERCENTAGES OF TOTAL NUMBER OF READS, NUMBER OF HOMOPOLYMER STRETCHES AND NUMBER OF BASES AFTER CORRECTION RELATIVE TO THE ORIGINAL DATA ARE GIVEN.

	Original data	***Tombo corrected data***	***Reference***
Total number of reads	147,861	81,703 (55.26%)	
Total number of homopolymer stretches	454,105	13,747,629 (207.94%)	19,247
Total number of bases in homopoly-mers	33,066,885	73,636,414 (222.69%)	103,097
Homopolymer content	0.66%	2.22%	2.22%

TABLE 4 - NETWORK HYPERPARAMETERS AND PERFORMANCE. THE HYPERPARAMETER SETTINGS PER NETWORK ARE PRESENTED HERE. KEEP PROBABILITY IS THE PROPORTION OF GRU UNITS TO KEEP WHILE APPLYING DROPOUT DURING TRAINING. NETWORK PERFORMANCE IS DESCRIBED BY THE MEASURES ACCURACY, PRECISION, RECALL AND F1. *ORIGINAL* INDICATES ON WHICH RNNS CERTAIN RESNETRNNS WERE BASED. NETWORKS ARE ORDERED BY F1 SCORE. BLUE: RNN; WHITE: RESNETRNN; GREEN: SELECTED NETWORKS FOR FURTHER TRAINING; RED: EXCLUDED NETWORKS; ASTERISKS INDICATE NETWORK WITH SAME HYPERPARAMETERS.

network	original	batch size	optimizer	learning rate	size of RNN layers	number of RNN layers	keep probability	size of ResNet layers	number of ResNet layers	accuracy	precision	recall	F1	
ResNetRNN	1	256	RMSProp	0.001	64	3	0.8	32	2	89.90	14.58	73.28	0.2432	
ResNetRNN	X	128	RMSProp	0.001	256	4	0.2	32	4	90.10	14.47	72.15	0.2411	
ResNetRNN	X	256	Adam	0.001	32	2	0.5	64	2	89.67	14.21	73.33	0.2381	
ResNetRNN	2	512	Adam	0.001	128	2	0.4	256	5	89.36	14.11	75.90	0.2380	
ResNetRNN	1	256	RMSProp	0.001	64	3	0.8	64	2	89.16	13.91	75.28	0.2348	
ResNetRNN	1	256	RMSProp	0.001	64	3	0.8	256	2	89.09	13.85	75.02	0.2338	
ResNetRNN	2	512	Adam	0.001	128	2	0.4	32	1	88.89	13.62	75.93	0.2310	
ResNetRNN	1	256	RMSProp	0.001	64	3	0.8	32	2	89.00	13.54	75.02	0.2294	
ResNetRNN	4	128	Adam	0.001	64	5	0.3	16	3	88.55	13.05	74.31	0.2220	
ResNetRNN	4	128	Adam	0.001	64	5	0.3	256	3	88.52	13.02	74.61	0.2217	
ResNetRNN	1	256	RMSProp	0.001	64	3	0.8	16	1	88.04	12.92	76.53	0.2211	
ResNetRNN	X	256	RMSProp	0.001	64	3	0.8	32	1	88.12	12.89	76.48	0.2206	
ResNetRNN	3	512	RMSProp	0.01	32	3	0.4	256	2	88.40	12.92	73.84	0.2199	
ResNetRNN	X	512	Adam	0.001	128	2	0.4	16	1	87.78	12.69	77.37	0.2180	
RNN	2	512	Adam	0.001	128	2	0.4	0	0	87.78	12.69	76.92	0.2179	
ResNetRNN	X	128	Adam	0.01	32	2	0.4	128	2	88.43	12.72	72.67	0.2165	
ResNetRNN	X	128	RMSProp	0.0001	128	2	0.6	32	2	88.16	12.58	73.32	0.2148	
ResNetRNN	3	512	RMSProp	0.01	32	3	0.4	32	4	88.45	12.61	71.91	0.2146	
ResNetRNN	2	512	Adam	0.001	128	2	0.4	16	3	87.44	12.45	77.29	0.2145	
RNN	1	256	RMSProp	0.001	64	3	0.8	0	0	87.75	12.47	75.32	0.2140	
ResNetRNN	2	512	Adam	0.001	128	2	0.4	32	4	87.44	12.36	77.85	0.2133	
ResNetRNN	4	128	Adam	0.001	64	5	0.3	128	1	87.43	12.32	76.34	0.2122	
RNN	3	512	RMSProp	0.01	32	3	0.4	0	0	87.73	12.34	74.54	0.2117	
ResNetRNN	3	512	RMSProp	0.01	32	3	0.4	64	3	88.68	12.47	68.04	0.2108	
ResNetRNN	2	512	Adam	0.001	128	2	0.4	16	2	87.07	12.14	78.13	0.2101	
ResNetRNN	X	128	RMSProp	0.001	256	2	0.3	64	2	86.92	12.11	78.47	0.2098	
ResNetRNN	4	128	Adam	0.001	64	5	0.3	32	3	87.23	12.03	76.09	0.2078	
ResNetRNN	X	128	Adam	0.001	16	5	0.8	128	5	87.07	11.83	75.57	0.2046	
RNN	4	128	Adam	0.001	64	5	0.3	0	0	86.79	11.77	76.06	0.2039	
ResNetRNN	X	512	RMSProp	0.0001	256	1	0.6	32	1	87.67	11.72	69.80	0.2007	
ResNetRNN	4	128	Adam	0.001	64	5	0.3	64	4	86.63	11.45	76.14	0.1991	
ResNetRNN	3	512	RMSProp	0.01	32	3	0.4	64	5	87.54	11.52	69.58	0.1977	
RNN	X	256	Adam	0.001	16	2	0.6	0	0	86.95	11.35	72.49	0.1963	
RNN	X	256	Adam	0.0001	128	3	0.5	0	0	86.65	11.32	73.08	0.1960	
RNN	X	128	RMSProp	0.001	32	4	0.5	0	0	85.85	11.13	76.30	0.1943	
RNN	X	128	RMSProp	0.001	32	5	0.4	0	0	85.82	11.12	76.68	0.1942	
RNN	X	512	RMSProp	0.0001	32	1	0.3	0	0	87.26	11.25	69.61	0.1937	*
ResNetRNN	X	512	RMSProp	0.0001	32	2	0.2	128	2	85.16	10.47	75.11	0.1838	
RNN	X	512	Adam	0.0001	16	2	0.8	0	0	85.86	10.54	71.51	0.1837	
RNN	X	512	RMSProp	0.001	16	1	0.4	0	0	86.71	10.62	67.55	0.1835	
RNN	X	128	Adam	0.001	16	4	0.5	0	0	85.18	10.44	75.44	0.1834	
ResNetRNN	3	512	RMSProp	0.01	32	3	0.4	64	3	86.51	10.49	67.78	0.1817	
RNN	X	256	RMSProp	0.0001	256	2	0.2	0	0	85.41	10.27	72.60	0.1799	
RNN	X	256	RMSProp	0.001	16	1	0.7	0	0	85.59	10.02	69.01	0.1750	
RNN	X	512	Adam	0.0001	16	2	0.8	0	0	85.14	9.74	69.45	0.1708	
RNN	X	256	Adam	0.0001	64	2	0.7	0	0	84.16	9.52	72.54	0.1683	
RNN	X	512	RMSProp	0.0001	32	1	0.3	0	0	85.42	9.40	64.71	0.1642	*
RNN	X	256	Adam	0.01	16	1	0.6	0	0	85.51	9.33	63.72	0.1628	
RNN	X	512	Adam	0.0001	32	4	0.5	0	0	82.39	8.90	75.14	0.1591	
RNN	X	256	Adam	0.0001	16	3	0.6	0	0	82.40	8.62	71.96	0.1540	
RNN	X	256	RMSProp	0.0001	16	3	0.8	0	0	83.28	8.29	71.01	0.1485	
RNN	X	128	Adam	0.01	32	4	0.6	0	0	85.02	8.26	57.19	0.1444	
RNN	X	512	Adam	0.1	64	5	0.5	0	0	81.01	7.53	67.11	0.1354	
RNN	X	128	RMSProp	0.0001	16	5	0.5	0	0	78.38	7.36	75.51	0.1341	
ResNetRNN	X	512	RMSProp	0.01	128	1	0.3	16	1	68.77	5.18	75.75	0.0970	
RNN	X	256	RMSProp	0.01	128	4	0.6	0	0	89.88	4.80	19.19	0.0768	
RNN	X	128	RMSProp	0.01	128	3	0.7	0	0	63.27	3.84	65.28	0.0725	
ResNetRNN	X	256	RMSProp	0.1	128	2	0.8	32	2	2.23	2.23	100.00	0.0436	
RNN	X	128	RMSProp	0.01	128	4	0.4	0	0	65.01	2.27	42.65	0.0431	
RNN	X	512	RMSProp	0.01	256	4	0.3	0	0	97.78	1.99	0.03	0.0006	
RNN	X	128	RMSProp	0.1	16	4	0.6	0	0	97.79	1.12	0.00	0	
ResNetRNN	X	512	RMSProp	0.1	256	5	0.3	16	5	97.80	0	0	0	
ResNetRNN	X	512	Adam	0.01	128	5	0.5	16	5	97.79	0	0	0	
ResNetRNN	X	256	Adam	0.1	128	3	0.3	128	3	97.80	0	0	0	
RNN	X	512	Adam	0.1	64	5	0.5	0	0	97.79	0	0	0	
RNN	X	128	Adam	0.1	256	3	0.3	0	0	0	0	0	0	
RNN	X	128	Adam	0.1	256	3	0.6	0	0	0	0	0	0	