UNIVERSITY OF ZAGREB
**FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING**

MASTER THESIS num. 1572

# End-to-End Deep Learning Model for Base Calling of MinION Nanopore Reads

Neven Miculinić

Zagreb, June 2018.

*Umjesto ove stranice umetnite izvornik Vašeg rada.*
*Kako biste uklonili ovu stranicu, obrišite naredbu* `\izvornik`.

*I would like to thank my mentor, Mile Šikić, for his patient guidance, encouragement and advice provided over the years.*

*I would also like to thank my family and friends for their continuous support.*

*In the end, honorable mentions go to Marko Ratković for his help with this thesis.*

# CONTENTS

# LIST OF FIGURES

# List of Tables

# 1. Introduction

In recent years, deep learning methods significantly improved the state-of-the-art in multiple domains such as computer vision, speech recognition and natural language processing (**??**) In this paper, we present application of deep learning for DNA basecalling problem.

Oxford Nanopore Technology's MinION nanopore sequencing platform **?** is the first portable DNA sequencing device. It produces longer reads than competing technologies. In addition, it enables real-time data analysis which makes it suitable for various applications. Although MinION is able to produce long reads, even up to 882 kb **??**, they have an error rate of 10% or higher. This master thesis uses R9.4 pore model and compares previous techniques with novel auto-encoder multi-task training.

## 1.1. Organization

[TODO]: Write some fancy stuff once completed.

# 2. Background

Due to technical constraints, it's infeasible to sequence whole DNA in single strand. Every sequencing technology to date have an upper limit how big strand can it precisely sequence. This limit is considerably smaller than size of genome. For example E.Coli has 4.5 million base pairs in its DNA, while Sanger's sequencing maximum output is around 1000 base pairs max. To make DNA basecalling feasible technique called shotgun sequencing was invented. The strand is cloned number of times, then via chemical agent broken down into smaller fragments of appropriate length. Sequenced fragments are called reads.

Genome assembly is the process of reconstructing the original genome from reads and usually starts with finding overlaps between reads. The quality of reconstruction heavily depends on the length and the quality (accuracy) of the reads produced by the sequencer.

If we have reference sequence we usually align the reads on the reference to aid us into genome assembly. Otherwise we have to use many de novo assembly techniques.

The right analogy would be building a puzzle. Since we cannot scan the whole puzzle because our camera is too small or imprecise, we are scanning pieces of the whole picture. Puzzle pieces would represent fragments in this analogy. If we have a map, even a rough one, it shall aids us into assemblying those puzzle pieces into complete pictures. Otherwise we're fiddling in the dark and using de novo assembly techniques.

Figure **??** depicts process of sequencing visually.

In 1977, Frederick Sanger (**?**)(**?**) started development of sequencing technologies. It allowed read lengths up to 1000 bases with very high accuracy(99.9%) at the cost of 1$ per 1000 bases. Later, seconds generation sequencing, like IAN Torrent and Illumina devices, reduced the price while keeping the accuracy high. However, they had a cost of shorter read lengths, about a few hundred base pairs, which makes resolving repetitive regions practically impossible.

Third generation sequencing technologies have longer read lengths at the accuracy's expense. PacBio, for example, developed technology with a few thousand bases with error rates of ~10-15%.

MinION sequences, which this master thesis use, made sequencing less expensive and even portable.

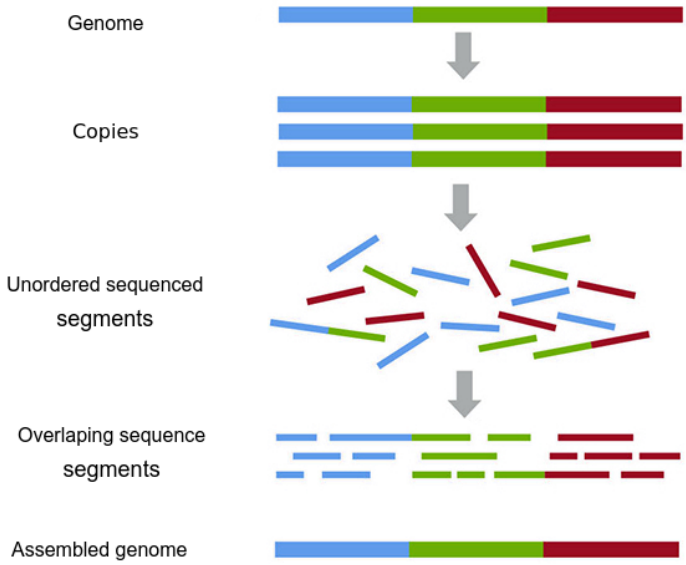


**Figure 2.1:** Depiction of the sequencing process

## 2.1. Oxford Nanopore MinION

The MinION device by Oxford Nanopore Technologies is the first portable DNA sequencing device. Its small weight, low cost, and long read length combined with decent accuracy yield promising results in various applications including full human genome assembly **?** what could potentially lead to personalized genomic medicine. It weights only 87 grams, and its portability lead to uses on international space station and the antarctic among other places.

Under the hood, it has numerous nano-meter sized pores, thus a names Nanopore. Each pore has width of around 6 nucleotied. Under electric current DNA strand passed through the pore and changes its electric resistance. The sensor measures current through the pore multiple times a second [1]. This signal varies depending which k-mer is occupying the pore, and on its basis we're performing the basecalling. On figure **??** this process is visually depicted.

MinION devices can produce long reads, usually tens of thousand base pairs (with reported reads lengths of 100 thousand **?** and even recently above 800 thousand base pairs **?**), but with high sequencing error than older generations of sequencing technologies.

The sequecing resulting file is in FAST5 format, which is adapted HDF5 file format, popular in bioinformatics community. It stores raw signal, alongside various metadata. Unfortunately, many basecallers, including the official ones, upon executing store their results in the FAST5 files. It leads to data and processing coupling in single file, and bloated file

[1] The model we worked with had 4000 samples per seconds

[2] Figure adapted from https://nanoporetech.com/how-it-works

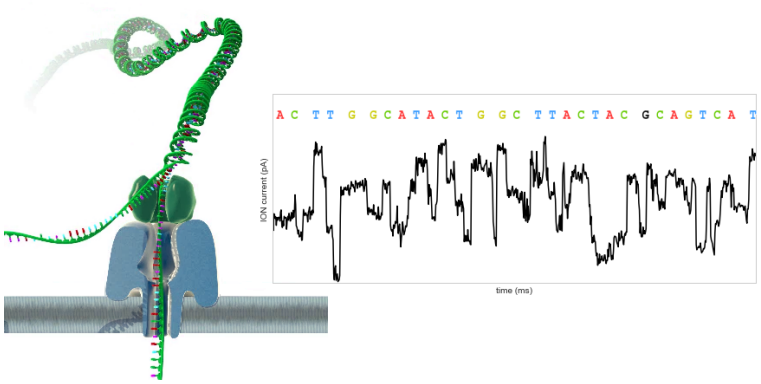


**Figure 2.2:** DNA strain being pulled through a nanopore [2]

sizes.

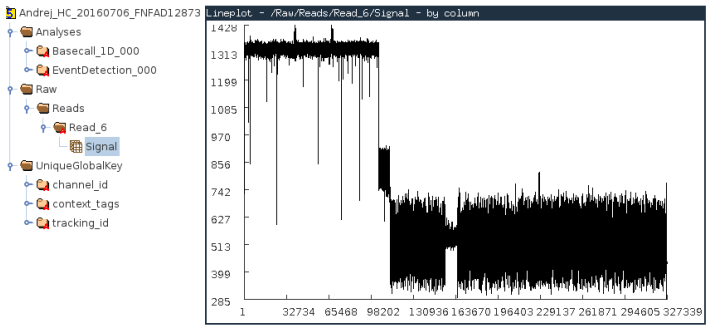


**Figure 2.3:** Structure of FAST5 file and raw signal line plot show in *HDFView* [3]

## 2.2. Related work

### 2.2.1. Oxford Nanopore Technologies

This subsection covers various basecallers published by ONT, the MinION device maker.

#### Metrichor

Metrichor is now defunct cloud based basecaller. The older versions used *hidden Markov models* (HMM) as underlying algorithm. Preprocessing started with segmenting signal into smaller chunks called events with their start, end, length, mean signal strength and standard deviation. This events are observations in the HMM model, and the underlying generating hidden sequence is sequence of 6-mers. They build their HMM transition matrix with stay,

[3] https://support.hdfgroup.org/products/java/hdfview/

skip 1 and skip 2 probabilities, that is underlying 6-mer cannot move for more than 2 nucleotides per event. Basecalling is performed using Viterbi algorithm. This approach showed poor results when calling long homopolymer stretches as the context in the pore remains the same **??**.

### Nanonet

Nanonet is ONT first generation neural network basecaller. It used to be available on github, but it's now defunct and unavailable.

### Albacore

Albacore is a production basecaller provided by Oxford Nanopore, and uses a command-line interface. It utilizes the latest in Recurrent Neural Network algorithms in order to interpret the signal data from the nanopore, and basecall the DNA or RNA passing through the pore. It implements stable features into Oxford Nanopore Technologies' software products, and is fully supported. It receives .fast5 files as an input, and is capable of producing:

- .fast5 files appended with basecalled information
- .fast5 files that have been processed, but basecall information present in a separate .fastq file

### Guppy

Guppy is ONT's new basecaller that can use GPUs to basecall much faster than Albacore. Both the GridION X5 and PromethION contain GPUs and use Guppy to basecall while sequencing. Guppy can also use CPUs and scales well to many-CPU systems, so it may run faster than Albacore even without GPUs. In the future it's intended to replace Albacore as production basecaller.

### Scrappie

Scrappie[4] is ONT's research basecaller. Scrappie is reported to be the first basecaller that specifically address homopolymer base calling. It became publicly available just recently in June, 2017 and supports R9.4 and future R9.5 data.

Unlike Albacore, Scrappie does not have fastq output, either directly or by writing it into the fast5 files – it only produces fasta reads.

[4] `https://github.com/nanoporetech/scrappie`

### 2.2.2. Third-party basecallers

#### Nanocall

Nanocall (**?**) was the first third-party open source basecaller for nanopore data. It uses HMM approach like the original R7 Metrichor. Nanocall does not support newer chemistries after R7.3.

#### DeepNano

DeepNano (**?**) was the first open-source basecaller based on neural networks. It uses bidirectional recurrent neural networks implemented in Python, using the Theano library. When released, originally only supported R7 chemistry, but support for R9 and R9.4 was added recently.

#### basecRAWller

basecRAWller [5] is developed by Marcus Stoiber and James Brown at the Lawrence Berkeley National Laboratory.

#### Chiron

Chiron (**?**) is developed by Haotian Teng and others in Lachlan Coin's group at the University of Queensland. They are basecalling from the raw signal, using first residual convolutional neural network, than LSTM and finally beam search or greedy decoder depending on chosen configuration.

[5] `https://basecrawller.lbl.gov/`

# 3. Methods

This chapter is dedicated to explaining key deep learning concepts used throughout the master thesis. It's here primarily for completeness, and it's author recommendation to go into detail via other sources, for example Deep learning book (**?**), google, or research papers cited for most of the techniques.. TensorFlow (**?**) and Keras (**?**) deep learning frameworks were used for implementation. For each deep learning concept I'll provide equivalent keras/tensorflow code whichever one is simpler and used throughout the codebase.

## 3.1. Neural Network

Feed-forward Neural network is the basic building block of any deep learning system. It's composition of multiple differentiable functions. Commonly we have input vector $x$, apply some linear transformation to it and add bias, and finally on the result some activation function. Details on common choices for actionvation function are in section **??**. In mathematical language $y = f(Ax + b)$ would be one layer of neural network transforming input $x$ into output $y$. Stacking those operation we get multiple layers, hence the word deep in deep learning. Simple 3-layer neural network is depicted in figure **??**.

$$
\begin{aligned}
y_1 &= f(A_1 x + b_1) \\
y_2 &= f(A_2 y_1 + b_2) \\
y &= f(A_3 y_2 + b_3)
\end{aligned}
$$

**Figure 3.1:** Simple three layer feed forward neural network

## 3.2. Activation functions

Most neural network operations are linear transformation. Composing multiple linear transformation we get new linear transformation. Thus have non-linear behavior we use the non-linear activation functions. Originally, the most popular choice was $\tanh$ and $\sigma(x) = \frac{1}{1+e^{-x}}$ activation functions. They are nice because of limited output domain.

However, other choices proved more effective, especially with deep neural network due to greater learning speeds, and to overcome gradient vanishing problem. Gradient vanishing refers to neural network gradient approaching zero as we back propagate through more and more layers. Exploding gradient is related phenomena in which gradient approaches infinity. Both present serious hampering to neural network training.

ReLU, The rectified linear unit, $f(x) = \max(0, x)$, is one hugely popular choice and decent baseline compared to other ReLU variants. ReLU greatly accelerates the convergence of stochastic gradient descent compared to $\sigma$ and $\tanh$ activation functions (**?**).

Furthermore, its calculation is drastically simpler then computing transcendental functions, like $\sigma$ or $\tanh$.

Over time, ReLU showed its downsides, called *dying ReLU*. It still saturates the gradients when it's 0, that is when $x <= 0$ giving no useful gradient to back propagate. Thus several ReLU variant have been proposed: PrRelu (**?**) in equation **??**, ELU (**?**) in equation **??**, and finally SeLU (**?**) in equation **??**. In Selu constants $\alpha$ and $\lambda$ are chosen in such a way that output gravitates towards normal distribution with zero mean and unit variance. Those constants are: $\lambda = 1.0507$ and $\alpha = 1.6732$. Code generating the function plot is displayed in figure **??**, and the plot is in figure **??**

$$PrELU(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha x & \text{otherwise} \end{cases} \tag{3.1}$$

$$ELU(x) = \begin{cases} x & \text{if } x > 0 \\ \exp(x) - 1 & \text{otherwise} \end{cases} \tag{3.2}$$

$$\text{selu}(x) = \lambda \begin{cases} x & \text{if } x > 0 \\ \alpha e^x - \alpha & \text{otherwise} \end{cases} \tag{3.3}$$

## 3.3. RNN

RNN, residual neural networks are one of the basic building blocks in sequence models. They are feed-forward neural netwoek spanned in time domain. The same neural network is applied to two inputs, hidden state and input at time $t$ and gives two outputs, new hidden

```
import tensorflow as tf
import keras
import seaborn as sns
import numpy as np
import matplotlib.pyplot as plt

for act in ["relu","selu", "elu"]:
    with tf.Graph().as_default():
        x = tf.placeholder(shape=(None, ), dtype=tf.float32)
        y = keras.layers.Activation(act)(x)

        with tf.Session() as sess:
            xx = np.linspace(-1, 1)
            yy = sess.run(y, feed_dict={
                x: xx
            })
            plt.plot(xx, yy, label=act)
            plt.legend()
```

**Figure 3.2:** Code generating the activation function plot. Also shows how tensorflow and keras could be used

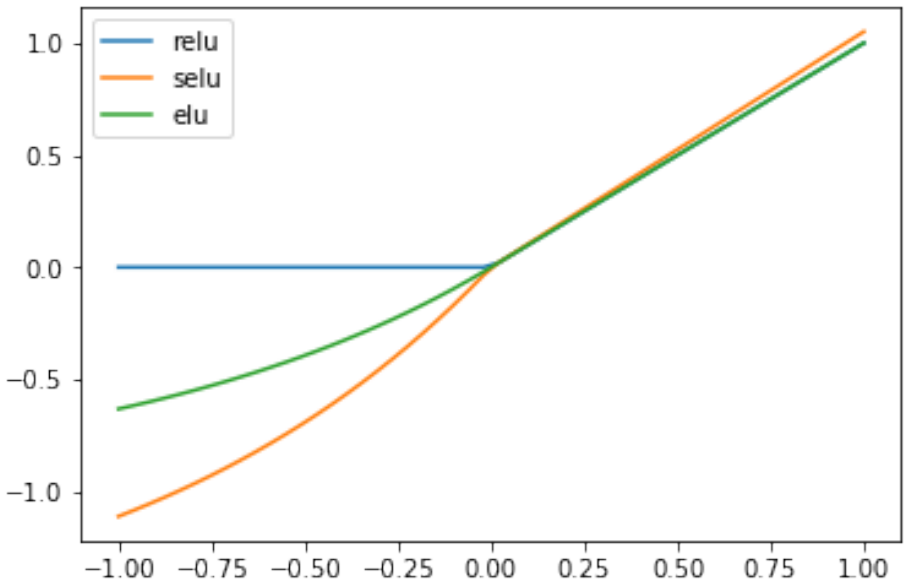


**Figure 3.3:** Plot of common activation functions between -1 and 1. PrELU is excluded since the parameter $\alpha$ is learnt during training

state and output at time $t$. The historic information remains saved in the hidden state, and it's propagated to the future. It's visual description is depicted in figure **??**.

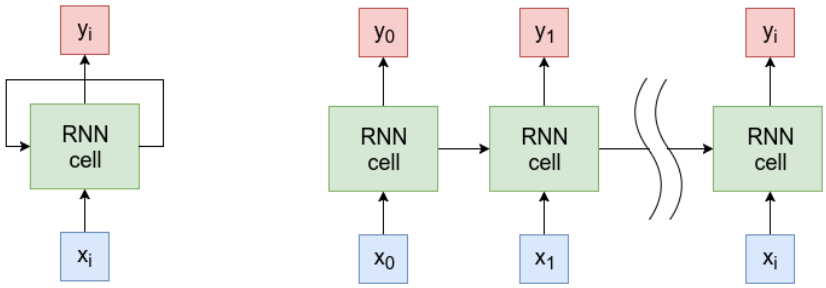


**Figure 3.4:** An unrolled recurrent neural network. Adapted from (**?**) with authors permission.

They can be unrolled in one massive feed-forward neural network. They are trained with backpropagation through time, on this unrolled graph. Usually the unrolling is capped and fixed number or time steps. Common issue to vanilla RNNs is vanishing and exploding gradients problem. It's solved by redesigning the basic RNN cell. There are two common approaches called LSTM (**?**) and GRU (**?**).

Bidirectional Recurrent Neural (BiRNN) networks are used when the current output not only depends on the previous elements in the sequence but also future elements. Basically we're stacking to RNNs, one in forward direction and another in the backward and concatenating the hidden states/outputs. This approach was used in DeepNano (**?**).

Despite their modeling power, main drawback is their speed. Since they are processing data sequentially, it's hard to paralize those operations.

# 4. System architecture

## 4.1. Data Preparation

Data has been downloaded from `https://data.genomicsresearch.org/Projects/online_dataset/train_set_all/`. The following species were provided there by the Chiron team:

- Human
- E. Coli
- Lambda Phage

The raw dataset is transformed using my repo, minion-data [1]. It defines common dataset training structure in the protobuf (**?**) interface description language (IDL). The whole definition can be seen in figure **??**.

For the concrete Chiron dataset, the re-squiggled preparation method was used. (The data was re-squiggled, that is after aligning the read on the reference, the read data is improved and each base pairs place on the raw signal is calculated.)

The re-squiggled basecalled data is located at `/Analyses/RawGenomeCorrected_000/BaseCa`
The interesting code fragments are in function processDataPoint of file `minion_data/preperation/_`
from minion-data python package.

After the gzipped dataset is prepared, it goes into the training pipeline. The whole training & testing pipeline is available open source on `https://github.com/nmiculinic/minion-basecaller`

## 4.2. Training pipeline

## 4.3. Hyperparameter optimization

[1] `https://github.com/nmiculinic/minion-data`

```
syntax = "proto3";

package dataset;

enum BasePair {
    A = 0;
    C = 1;
    G = 2;
    T = 3;
    BLANK = 4;
}

enum Cigar {
    MATCH = 0;
    MISMATCH = 1;
    INSERTION = 2; // Insertion, soft clip, hard clip
    DELETION = 3;  // Deletion, N, P
}

message DataPoint {
    message BPConfidenceInterval {
        uint64 lower = 1;
        uint64 upper = 2;
        BasePair pair = 3;
    }
    repeated float signal = 1;
    repeated BasePair basecalled = 2; // What we basecalled
    repeated BPConfidenceInterval labels = 3; // labels describe corrected basecalled signal for training
}

```

**Figure 4.1:** dataset protobuf description

# 5. Results

# 6. Conclusion

# BIBLIOGRAPHY

Yann LeCun and Yoshua Bengio. The handbook of brain theory and neural networks. chapter Convolutional Networks for Images, Speech, and Time Series, pages 255–258. MIT Press, Cambridge, MA, USA, 1998. ISBN 0-262-51102-9. URL http://dl.acm.org/citation.cfm?id=303568.303704.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 1097–1105. Curran Associates, Inc., 2012. URL https://goo.gl/UpFBv8.

Alexander S Mikheyev and Mandy MY Tin. A first look at the oxford nanopore minion sequencer. *Molecular ecology resources*, 14(6):1097–1102, 2014.

Nick Loman. *Nanopore R9 rapid run data release*, a. URL http://lab.loman.net/2016/07/30/nanopore-r9-data-release/. [Online; posted 30-July-2016].

Nick Loman. *Thar she blows! Ultra long read method for nanopore sequencing*, b. URL http://lab.loman.net/2017/03/09/ultrareads-for-nanopore/. [Online; posted 9-March-2017].

Mirjana Domazet-Lošo Mile Šikić. *Bioinformatika*. Bioinformatics - course matherials, Faculty of Electrical Engineering and Computing, University of Zagreb, 2013.

Erik Pettersson, Joakim Lundeberg, and Afshin Ahmadian. Generations of sequencing technologies. *Genomics*, 93(2):105–111, feb 2009. doi: 10.1016/j.ygeno.2008.10.003. URL https://doi.org/10.1016/j.ygeno.2008.10.003.

Miten Jain, Sergey Koren, Josh Quick, Arthur C Rand, Thomas A Sasani, John R Tyson, Andrew D Beggs, Alexander T Dilthey, Ian T Fiddes, Sunir Malla, Hannah Marriott, Karen H Miga, Tom Nieto, Justin O'Grady, Hugh E Olsen, Brent S Pedersen, Arang Rhie, Hollian Richardson, Aaron Quinlan, Terrance P Snutch, Louise Tee, Benedict Paten, Adam M. Phillippy, Jared T Simpson, Nicholas James Loman, and Matthew Loose. Nanopore sequencing and assembly of a human genome with ultra-long reads. *bioRxiv*, 2017. doi:

10.1101/128835. URL http://biorxiv.org/content/early/2017/04/20/128835.

Sara Goodwin, John D. McPherson, and W. Richard McCombie. Coming of age: ten years of next-generation sequencing technologies. *Nat Rev Genet*, 17(6):333–351, Jun 2016. ISSN 1471-0056. URL http://dx.doi.org/10.1038/nrg.2016.49. Review.

Camilla L.C. Ip, Matthew Loose, John R. Tyson, Mariateresa de Cesare, Bonnie L. Brown, Miten Jain, Richard M. Leggett, David A. Eccles, Vadim Zalunin, John M. Urban, Paolo Piazza, Rory J. Bowden, Benedict Paten, Solomon Mwaigwisya, Elizabeth M. Batty, Jared T. Simpson, Terrance P. Snutch, Ewan Birney, David Buck, Sara Goodwin, Hans J. Jansen, Justin O'Grady, and Hugh E. Olsen and. Minion analysis and reference consortium: Phase 1 data release and analysis. *F1000Research*, oct 2015. doi: 10.12688/f1000research.7201.1. URL https://doi.org/10.12688/f1000research.7201.1.

Matei David, Lewis Jonathan Dursi, Delia Yao, Paul C Boutros, and Jared T Simpson. Nanocall: An open source basecaller for oxford nanopore sequencing data. *bioRxiv*, 2016. doi: 10.1101/046086. URL http://biorxiv.org/content/early/2016/03/28/046086.

Vladimír Boža, Broňa Brejová, and Tomáš Vinař. DeepNano: Deep recurrent neural networks for base calling in MinION nanopore reads. *PLOS ONE*, 12(6):e0178751, jun 2017. doi: 10.1371/journal.pone.0178751. URL https://doi.org/10.1371/journal.pone.0178751.

Haotian Teng, Minh Duc Cao, Michael B. Hall, Tania Duarte, Sheng Wang, and Lachlan Coin. Chiron: Translating nanopore raw signal directly into nucleotide sequence using deep learning. *bioRxiv*, 2017. doi: 10.1101/179531. URL https://www.biorxiv.org/content/early/2017/09/12/179531.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu,

and Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

François Chollet et al. Keras. https://keras.io, 2015.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification, 2015.

Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. Fast and accurate deep network learning by exponential linear units (elus), 2015.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks, 2017.

Marko Ratković. Deep learning model for base calling of minion nanopore reads, 2017.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9 (8):1735–1780, 1997.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014. URL http://arxiv.org/abs/1412.3555.

Google. Protocol buffers. https://github.com/google/protobuf.

**End-to-End Deep Learning Model for Base Calling of MinION Nanopore Reads**

**Abstract**

In the MinION device, single-stranded DNA fragments move through nanopores, which causes drops in the electric current. The electric current is measured at each pore several thousand times per second. Each event is described by the mean and variance of the current and by event duration. This sequence of events is then translated into a DNA sequence by a base caller. Develop a base-caller for MinION nanopore sequencing platform using a deep learning architecture such as convolutional neural networks and recurrent neural networks. Instead of events, use current waveform at the input. Compare the accuracy with the state-of-the-art basecallers. For testing purposes use publicly, available datasets and Graphmap or Minimap 2 tools for aligning called reads on reference genomes. Implement method using TensorFlow or similar library. The code should be documented and hosted on a publicly available Github repository.

**Keywords:** base calling, Oxford Nanopore Technologies, MinION, deep learning, seq2seq, convolutional neural network, residual network, CTC loss

**S kraja na kraj model dubokog učenja za određivanje očitanih baza dobivenih uređajem za sekvenciranje MinION**

**Sažetak**

Unutar uređaja MinION, fragmenti jednostruke DNA prolaze kroz nanopore, što uzrokuje promjene u električnoj struji. Struja proizvedena na svakoj nanopori mjeri se nekoliko tisuća puta u sekundi. Svaki događaj opisan je srednjom vrijednosti i varijancom struje te svojim trajanjem. Postupak kojim se takav slijed događaja prevodi u niz nukleotida naziva se određivanje očitanih baza. Razviti alat za prozivanje baza za uređaj za sekvenciranje MinION koristeći modele dubokog učenje kao što su konvolucijske i povratne neuronske mreže. Umjesto događaja na ulazu koristi valni oblik struje. Usporediti dobivenu točnost s postojećim rješenjima. U svrhu testiranja koristiti javno dostupne skupove podataka i alate GraphMap ili Minimap 2 za poravnanje očitanja na referentni genom. Alat implementirati koristeći programsku biblioteku TensorFlow (ili neku sličnu). Programski kod treba biti dokumentiran i javno dostupan preko repozitorija GitHub. **Ključne riječi:** određivanje

baza, Oxford Nanopore Technologies, MinION, duboko učenje, prevođenje, konvolucijske neuronske mreže, rezidualne mreže, CTC gubitak