Deep Learning Model for Base Calling of MinION Nanopore Reads

Marko Ratković
Associate Profesor Mile Šikić, PhD

University of Zagreb
Faculty of Electrical Engineering and Computing

MinION

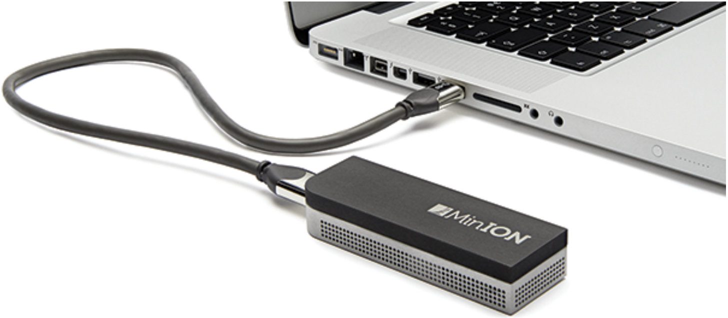


Figure 1: The MinION sequencing device

Technology

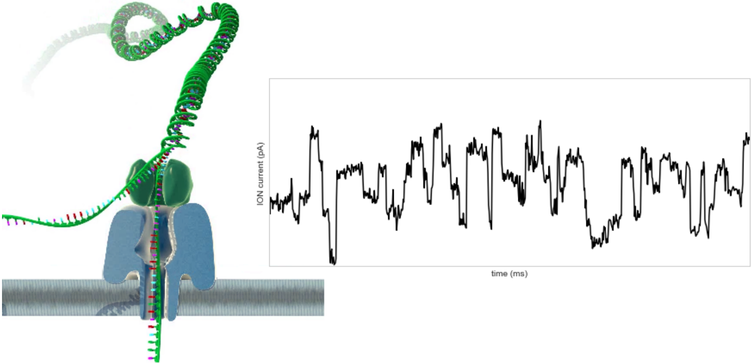
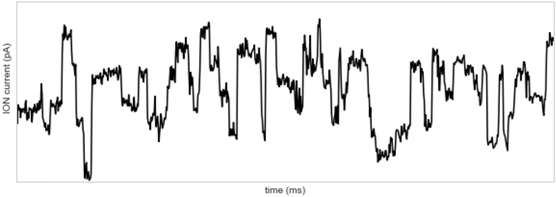
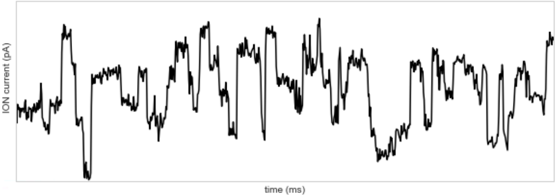


Figure 2: DNA strain being pulled through a nanopore

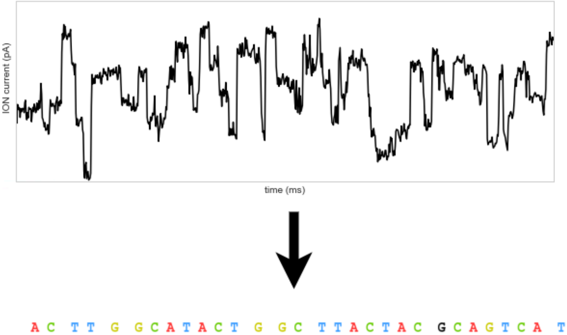
Basecalling



Basecalling



Basecalling



Basecalling options

Metrichor

- only basecaller for ONT data
- proprietary software
- available as a cloud service

Basecalling options

Metrichor

- only basecaller for ONT data
- proprietary software
- available as a cloud service

Goals

- local basecalling
- open-source
- speed, accuracy

Existing solutions

- Third-party: *DeepNano, NanoCall*
- Official: *MinKNOW, Nanonet, Albacore, Scrappie*

Existing solutions

- Third-party: *DeepNano, NanoCall*
- Official: *MinKNOW, Nanonet, Albacore, Scrappie*

Idea?

- Signal segmentation – event detection

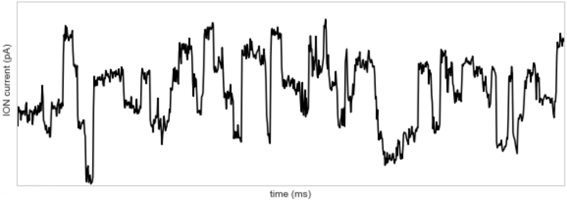


Figure 3: Signal segmentation

Existing solutions

- Third-party: *DeepNano, NanoCall*
- Official: *MinKNOW, Nanonet, Albacore, Scrappie*

Idea?

- Signal segmentation – event detection

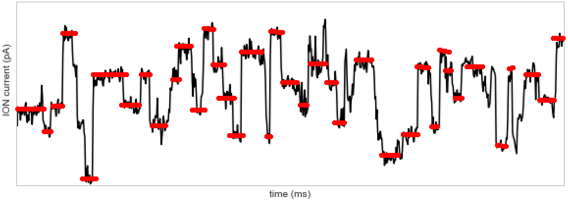


Figure 3: Signal segmentation

Existing solutions

- Third-party: *DeepNano, NanoCall*
- Official: *MinKNOW, Nanonet, Albacore, Scrappie*

Idea?

- Signal segmentation – event detection
- RNN, HMM (older version of *Metrichor* and *NanoCall*)

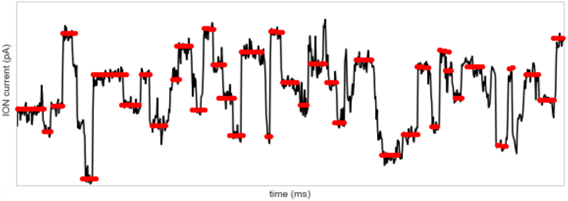


Figure 3: Signal segmentation

Proposed solution

end2end

Models

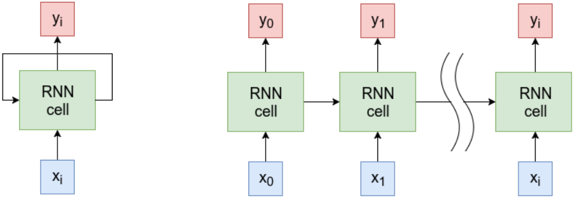


Figure 4: RNN

Models

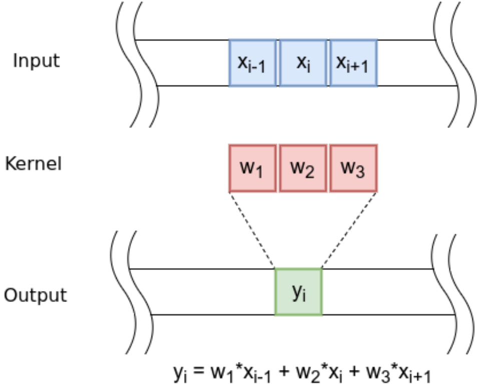


Figure 5: CNN

Models

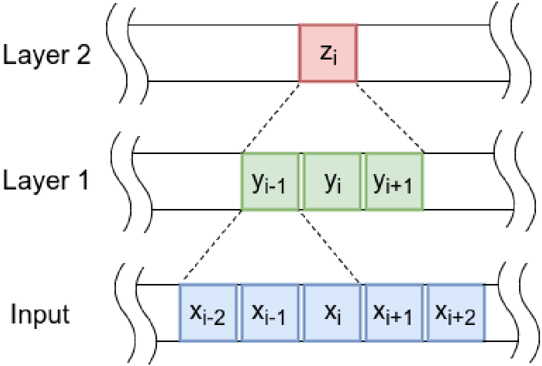


Figure 6: Receptive field

CTC loss

Idea: decode sequence from fixed-width output (softmax over alphabet)

CTC loss

Idea: decode sequence from fixed-width output (softmax over alphabet)

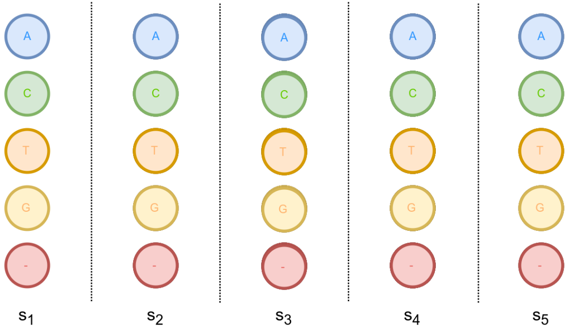


Figure 7: Output

CTC loss

Idea: decode sequence from fixed-width output (softmax over alphabet)

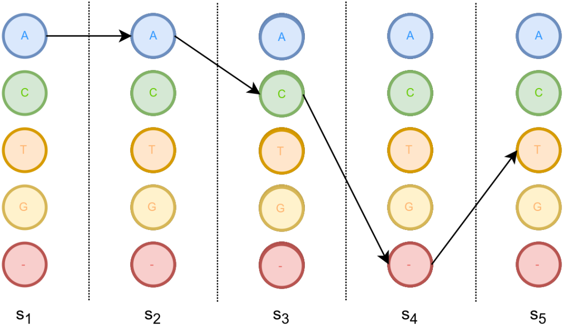


Figure 8: Path "AAC-T"

$$P(\pi|X) = \prod_{t=1}^{m} s_t(\pi_t) \tag{1}$$

CTC loss

Idea: decode sequence from fixed-width output

CTC loss

Idea: decode sequence from fixed-width output

$$ACT = \begin{cases} decode(A, A, A, C, T) \\ decode(A, A, C, -, T) \\ decode(-, A, C, T, T) \\ decode(-, -, A, C, T) \\ decode(A, C, C, C, T) \\ \vdots \\ decode(A, C, T, -, -) \end{cases} \tag{2}$$

CTC loss

Idea: decode sequence from fixed-width output

$$ACT = \begin{cases} decode(A, A, A, C, T) \\ decode(A, A, C, -, T) \\ decode(-, A, C, T, T) \\ decode(-, -, A, C, T) \\ decode(A, C, C, C, T) \\ \vdots \\ decode(A, C, T, -, -) \end{cases} \tag{2}$$

$$P(Y|X) = \sum_{\pi \in decode^{-1}(Y)} P(\pi|X) \tag{3}$$

CTC loss

Given the dataset $D = \{(X_i, Y_i)\}$, training objective is the maximization of the likelihood of each training sample which is the same as the minimization of negative log likelihood:

$$L(D) = -\sum_{(X,Y)\in D} lnP(Y|X) \tag{4}$$

Training data

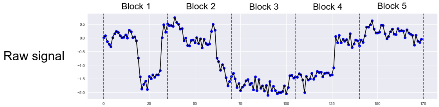


Figure 9: Dataset preparation

Training data

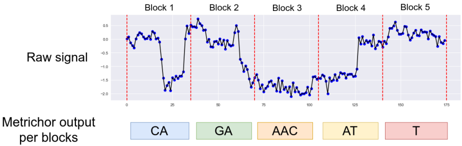


Figure 9: Dataset preparation

Training data

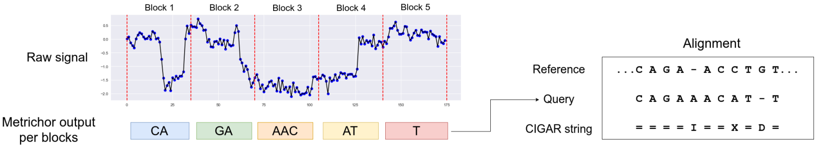
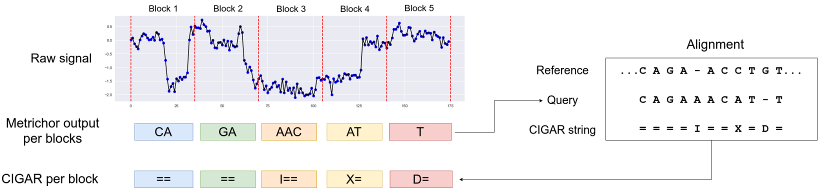


Figure 9: Dataset preparation

Training data



Block 1 Block 2 Block 3 Block 4 Block 5

Raw signal

Metrichor output per blocks

CA GA AAC AT T

CIGAR per block

== == I== X= D=

Alignment

Reference ...C A G A - A C C T G T...

Query C A G A A A C A T - T

CIGAR string = = = = I = = X = D =

Figure 9: Dataset preparation

Training data

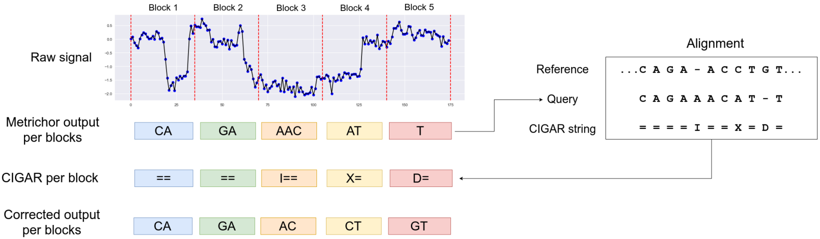


Figure 9: Dataset preparation

Status

- Loss
- Dataset
- Model? CNN?

Model

- Residual CNN

Model

- Residual CNN
- 72 blocks, 2M parameters

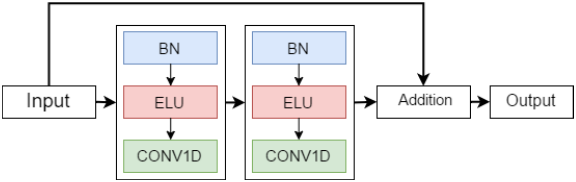


Figure 10: Residual block

Model

- Residual CNN
- 72 blocks, 2M parameters
- Maxpool every 24 blocks, reduction of dimensionality by factor 8

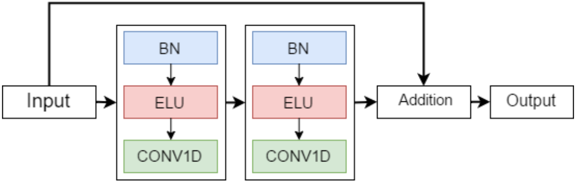


Figure 10: Residual block

Results

Results

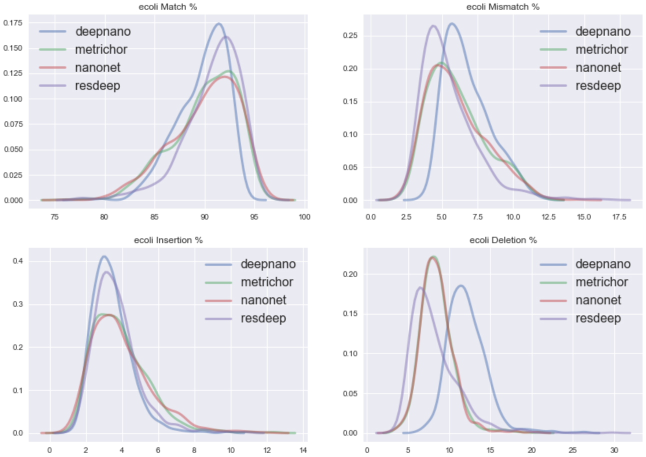


Figure 11: KDE plot of alimnment operations for E. Coli

Results

Table 1: Alignment specifications of E. Coli R9 basecalled reads

	Match % (median)	**Mismatch % (median)**	**Insertion % (median)**	**Deletion % (median)**
DeepNano	90.254762	6.452852	**3.274420**	11.829965
Metrichor	90.560455	5.688105	3.660381	8.328271
Nanonet	90.607674	5.608912	3.652791	8.299046
resdeep	**91.408591**	**5.019141**	3.477739	**7.471608**

Results

Table 2: Ecoli R9 basecalled read lengths in base pairs

	median	**mean**	**std**
DeepNano	5526.5	8126.694000	7406.554786
Metrichor	5809.5	8933.275000	9189.709720
Nanonet	3286.5	4874.406582	4803.182344
resdeep	5784.0	8990.988989	9297.972688

Results

Consensus from pileup

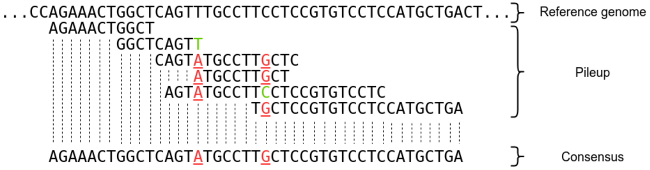


Figure 12: Consensus from pileup

Results

Consensus from pileup

Table 3: Consensus specification of E. Coli R9 basecalled reads

	Match %	**Snp %**	**Insertion %**	**Deletion %**
DeepNano	98.8742	1.0044	0.1214	0.9041
Metrichor	99.1223	0.7464	0.1313	0.6300
Nanonet	97.9691	1.5700	0.4609	1.5158
resdeep	**99.2361**	**0.6474**	**0.1165**	**0.5510**

Results

Consensus from de novo assembly obtained using assembler *ra*[1] and compared to the reference using *dnadiff* present in the Mumer[2]

[1] https://github.com/rvaser/ra

[2] https://github.com/garviz/MUMmer

Results

Consensus from de novo assembly obtained using assembler *ra*[1] and compared to the reference using *dnadiff* present in the Mumer[2]

Table 4: Assembly and consensus results for E. Coli

	Metrichor	**resdeep**	**Nanonet**
Aln. bases ref. (bp)	4639641(100.00%)	4639612(100.00%)	4639031(99.99%)
Aln. bases query (bp)	4604787(100.00%)	4614351(100.00%)	4599745(99.99%)
Avg. Identity	98.76	**99.06**	98.47
Edit distance	60418	**46686**	74341

[1] `https://github.com/rvaser/ra`

[2] `https://github.com/garviz/MUMmer`

Results - basecalling speed

Table 5: Base calling speeds measured in *base pairs per second*[3]

	resdeep	**Nanonet**	**DeepNano**
Speed CPU(bp/s)	**1363.34**	897.49	692.37
Speed GPU(bp/s)	**6571.76**	3828.39	-

[3] Tested on server with two 8-core Intel(R) Xeon(R) E5-2640 v2 CPUs, NVIDIA Titan Black GPU, 6GB

Conclusion

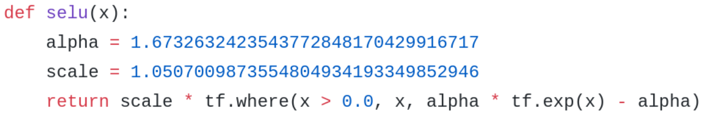
- end2end basecaller, open-source, fast and accurate
- Note: R9 data only, R9.4 in progress

Future work

- Scaled Exponential Linear Units (SELU), Jun 2017

Future work

- Scaled Exponential Linear Units (SELU), Jun 2017



```
def selu(x):
    alpha = 1.6732632423543772848170429916717
    scale = 1.0507009873554804934193349852946
    return scale * tf.where(x > 0.0, x, alpha * tf.exp(x) - alpha)
```

Figure 12: SELU implementation

Future work

- Scaled Exponential Linear Units (SELU), Jun 2017
- Facebook AI Research (FAIR) team: *Convolutional Sequence to Sequence Learning*, May 2017

Future work

- Scaled Exponential Linear Units (SELU), Jun 2017
- Facebook AI Research (FAIR) team: *Convolutional Sequence to Sequence Learning*, May 2017
- ...

Questions?