# 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

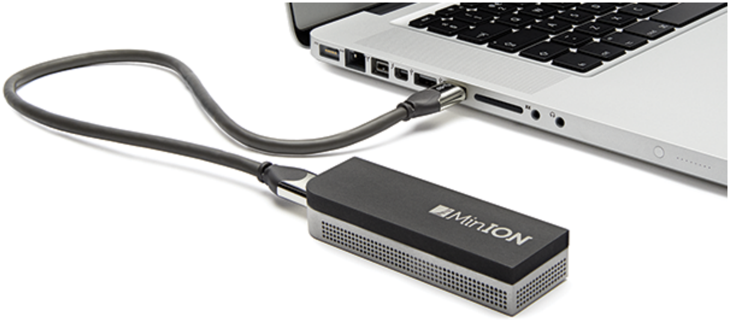


**Figure 1:** The MinION sequencing device

## Technology

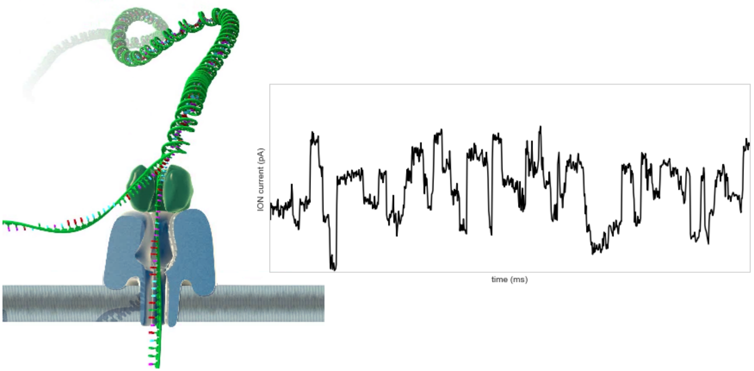
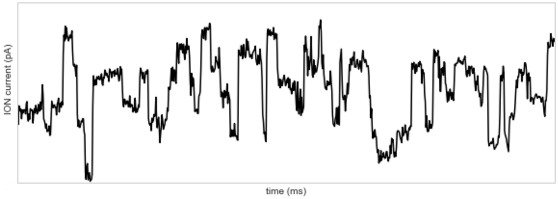
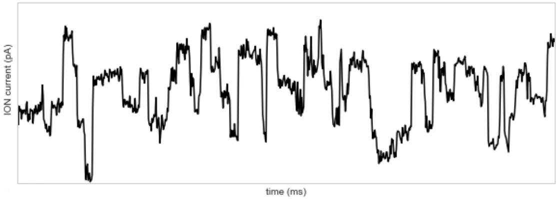


**Figure 2:** The MinION sequencing device

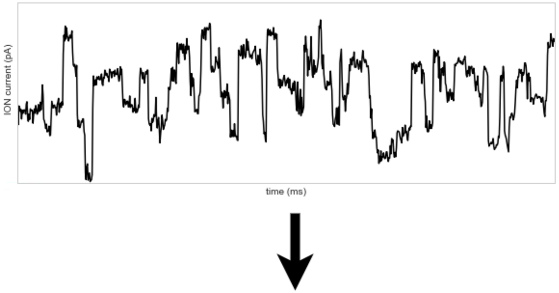
## Basecalling



## Basecalling



## Basecalling



A C  T T  G  G C A T A C T  G  G C  T T A C T A C  G C A G T C A  T

## 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*

Idea?

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

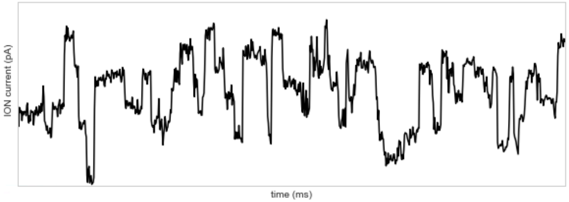


**Figure 3:** Signal segmentation

## Existing solutions

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

Idea?

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

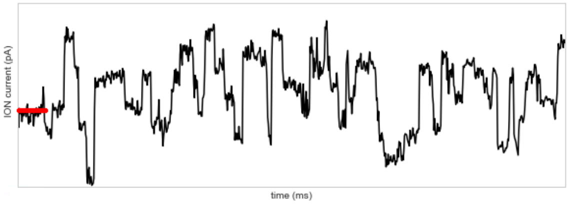


**Figure 3:** Signal segmentation

## Existing solutions

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

Idea?

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

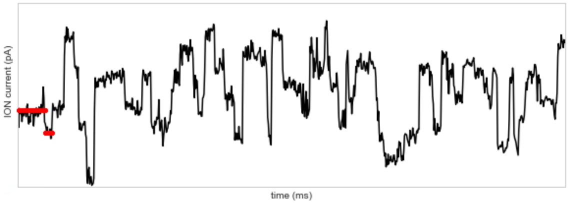


**Figure 3:** Signal segmentation

## Existing solutions

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

Idea?

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

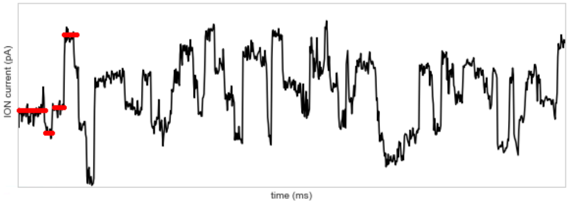


**Figure 3:** Signal segmentation

## Existing solutions

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

Idea?

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

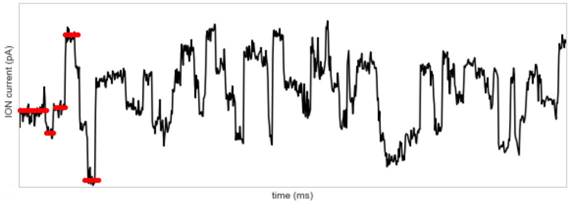


**Figure 3:** Signal segmentation

## Existing solutions

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

Idea?

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

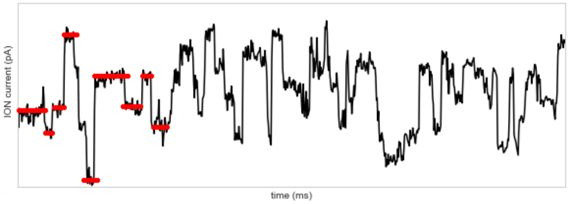


**Figure 3:** Signal segmentation

## Existing solutions

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

Idea?

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

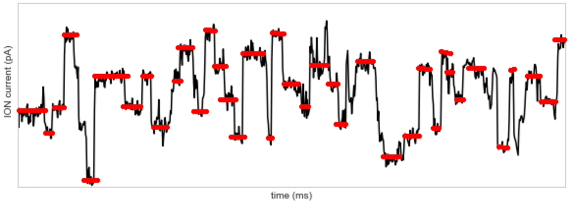


**Figure 3:** Signal segmentation

## Proposed solution

end2end, CNN, CTC loss
speed, paralelization, sequental, eliminate shit
variable length loss function

## CTC loss

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

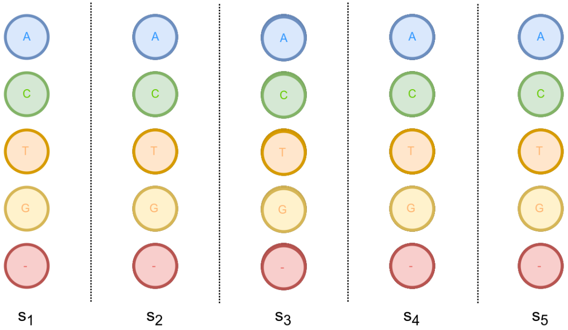


**Figure 4:** Output

## CTC loss

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

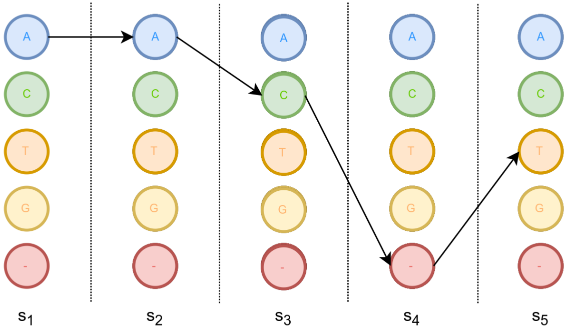


**Figure 5:** Path "AAC-T"

## CTC loss

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

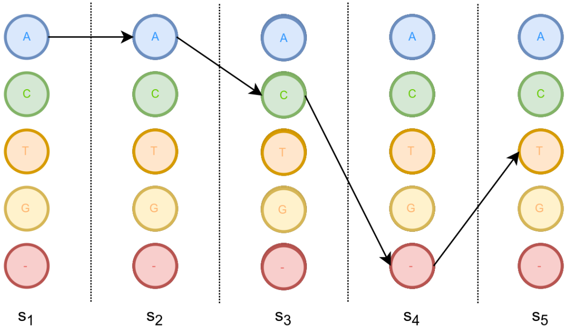


**Figure 5:** Path "AAC-T"

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

## 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

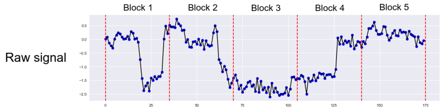


**Figure 6:** Dataset preparation

## Training

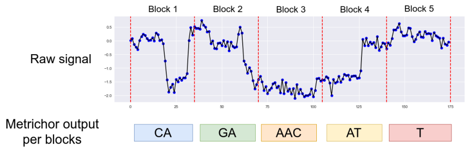


**Figure 6:** Dataset preparation

## Training

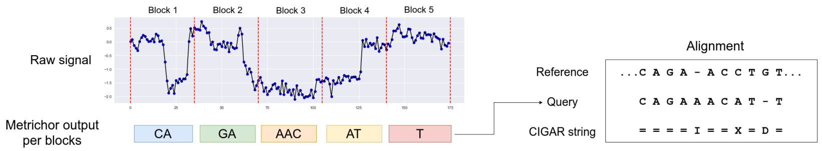


**Figure 6:** Dataset preparation

## Training

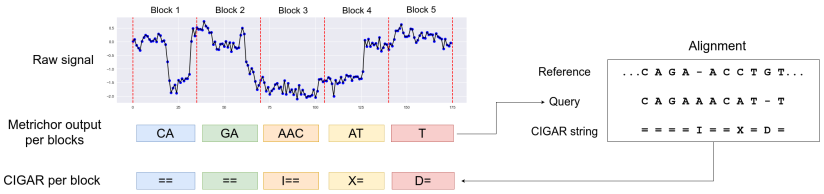


**Figure 6:** Dataset preparation

## Training

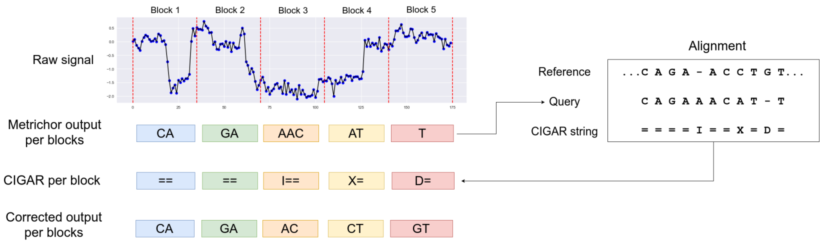


**Figure 6:** Dataset preparation

## Model

- Residual CNN

## Model

- Residual CNN
- 72 blocks, 2M parameters

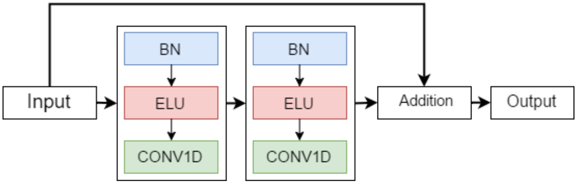


**Figure 7:** Residual block

## Model

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

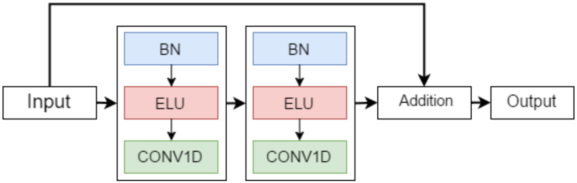


**Figure 7:** Residual block

## Results

Ground truth? Per read metrics

**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 |
| MinCall | **91.408591** | **5.019141** | 3.477739 | **7.471608** |

## Results

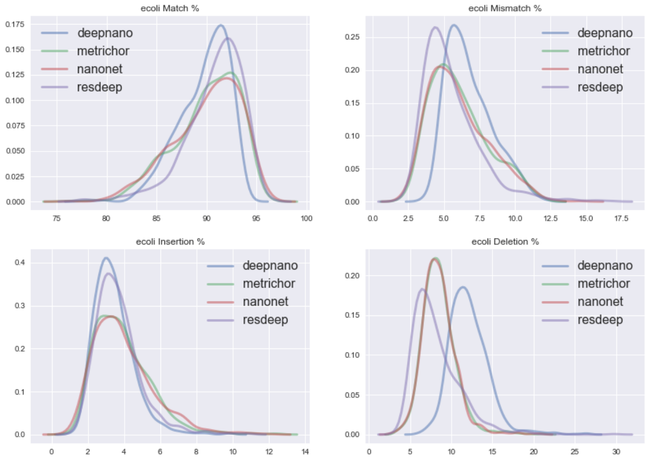


**Figure 8:** KDE plot of alimnment operations for E. Coli

## Results

### Consensus from pileup

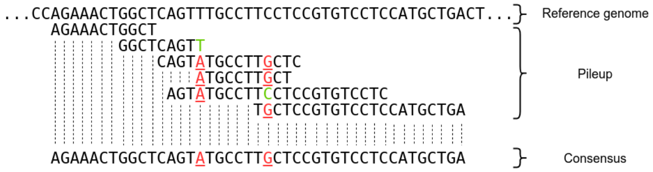


**Figure 9:** Consensus from pileup

## Results

### Consensus from pileup

**Table 2:** Consensus specification of E. Coli R9 basecalled reads

| | **Total called [bp]** | **Correctly called [bp]** | **Match %** | **Snp %** | **Insertion %** | **Deletion %** |
|---|---|---|---|---|---|---|
| DeepNano | 1510244.0 | 1493242.0 | 98.8742 | 1.0044 | 0.1214 | 0.9041 |
| Metrichor | 1515893.0 | 1502588.0 | 99.1223 | 0.7464 | 0.1313 | 0.6300 |
| Nanonet | 1414237.0 | 1385515.0 | 97.9691 | 1.5700 | 0.4609 | 1.5158 |
| MinCall | 1517828.0 | 1506233.0 | **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 3:** 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 4:** 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 X Black GPU, 6GB

## Future work

- Scaled Exponential Linear Units (SELU), Jun 2017

## Future work

- Scaled Exponential Linear Units (SELU), Jun 2017

```python
def selu(x):
    with ops.name_scope('elu') as scope:
        alpha = 1.6732632423543772848170429916717
        scale = 1.0507009873554804934193349852946
        return scale*tf.where(x>=0.0, x, alpha*tf.nn.elu(x))
```

**Figure 9:** 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
- ...

## End

Thank you for your attention!

## End

Thank you for your attention!

Any questions?