

# Deep Learning Model for Base Calling of MinION Nanopore Reads

---

Marko Ratković

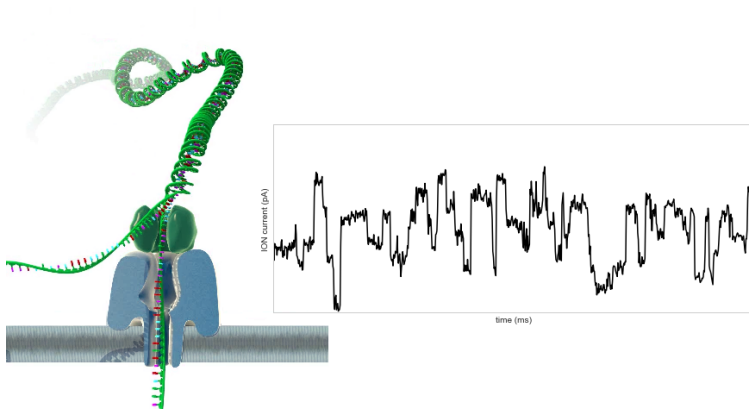
Associate Profesor Mile Šikić, PhD

University of Zagreb

Faculty of Electrical Engineering and Computing

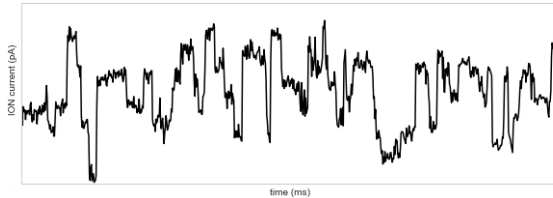


**Figure 1:** The MinION sequencing device

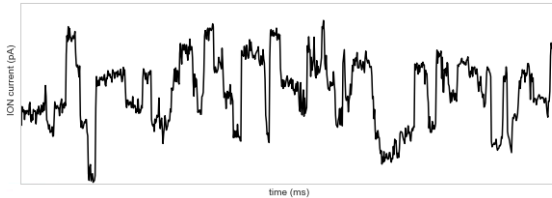


**Figure 2:** The MinION sequencing device

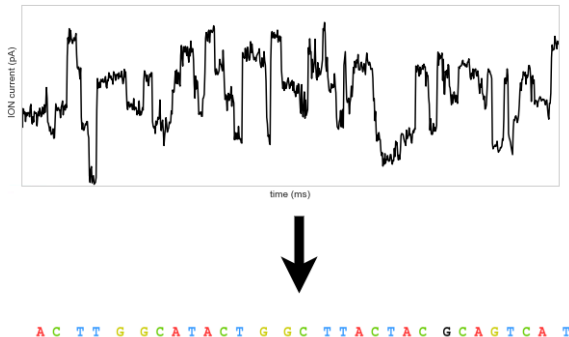
# Basecalling



# 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

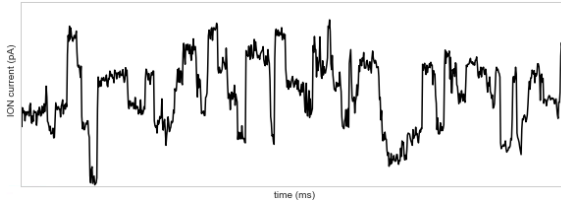
- Third-party: *DeepNano*, *NanoCall*
- Official: *MinKNOW*, *Nanonet*, *Albacore*, *Scrappie*

# Existing solutions

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

Idea?

- Signal segmentation – event detection



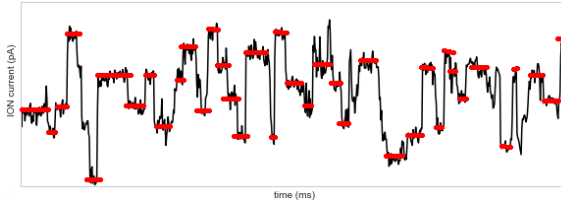
**Figure 3:** Signal segmentation

# Existing solutions

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

Idea?

- Signal segmentation – event detection



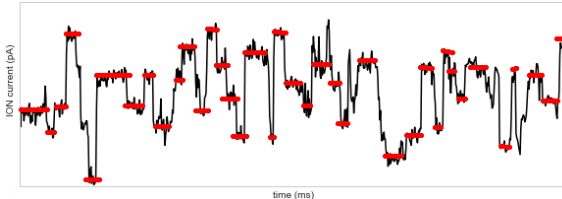
**Figure 3:** Signal segmentation

# Existing solutions

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

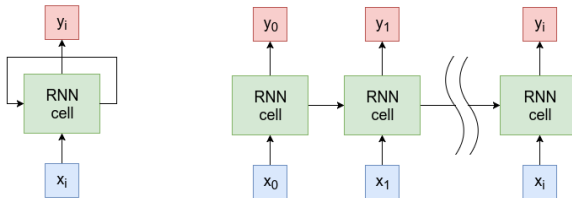
## Idea?

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

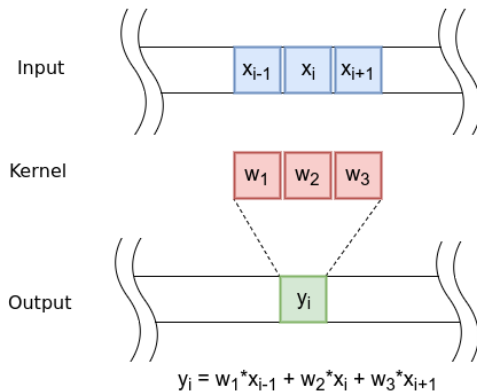


**Figure 3:** Signal segmentation

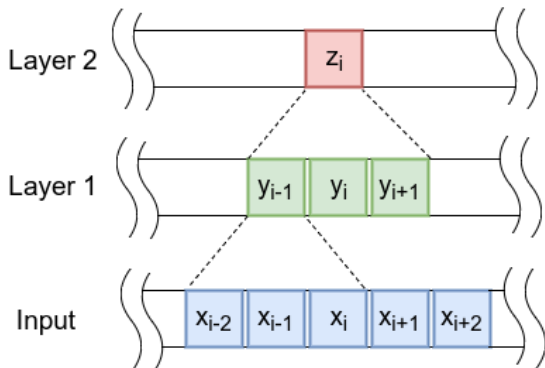
end2end



**Figure 4:** RNN



**Figure 5:** CNN



**Figure 6:** Receptive field



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

# CTC loss

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

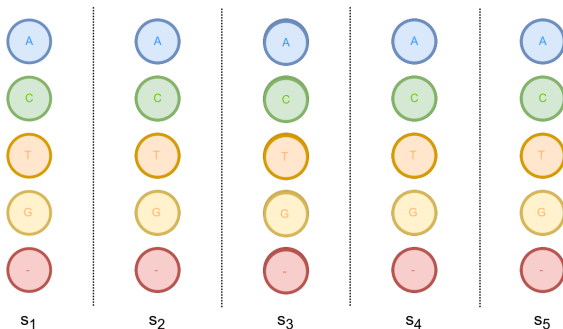


Figure 7: Output

# CTC loss

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

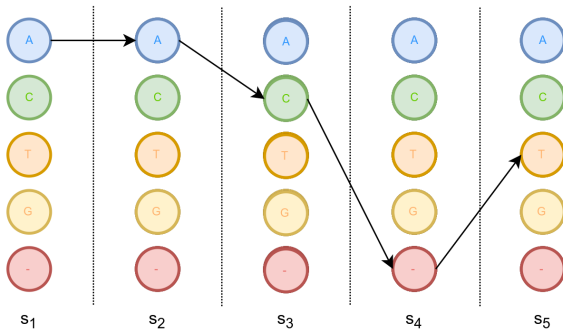


Figure 8: Path "AAC-T"

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

Idea: decode sequence from fixed-width output

Idea: decode sequence from fixed-width output

$$ACT = \left\{ \begin{array}{l} \text{decode}(A, A, A, C, T) \\ \text{decode}(A, A, C, -, T) \\ \text{decode}(-, A, C, T, T) \\ \text{decode}(-, -, A, C, T) \\ \text{decode}(A, C, C, C, T) \\ \vdots \\ \text{decode}(A, C, T, -, -) \end{array} \right. \quad (2)$$

Idea: decode sequence from fixed-width output

$$ACT = \left\{ \begin{array}{l} decode(A, A, A, C, T) \\ decode(A, A, C, -, T) \\ decode(-, A, C, T, T) \\ decode(-, -, A, C, T) \\ decode(A, C, C, C, T) \\ \vdots \\ decode(A, C, T, -, -) \end{array} \right. \quad (2)$$

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

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:

$$L(D) = - \sum_{(X,Y) \in D} \ln P(Y|X) \quad (4)$$

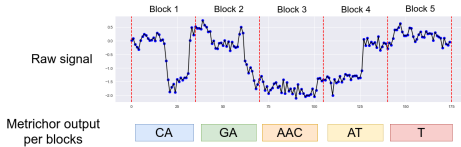
# Training data



**Figure 9:** Dataset preparation



# Training data



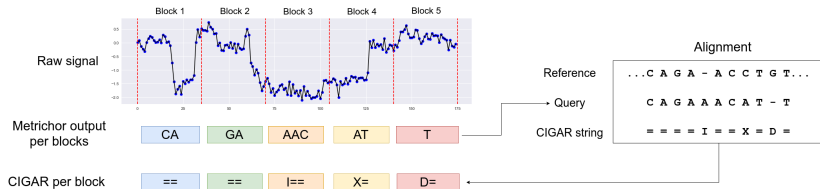
**Figure 9:** Dataset preparation

# Training data



**Figure 9:** Dataset preparation

# Training data



**Figure 9:** Dataset preparation

# Training data

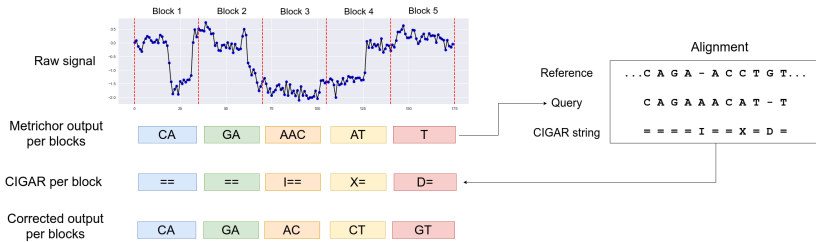


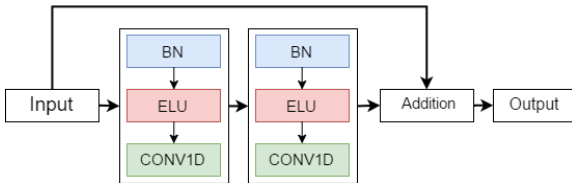
Figure 9: Dataset preparation

- Loss
- Dataset
- Model? CNN?

- Residual CNN

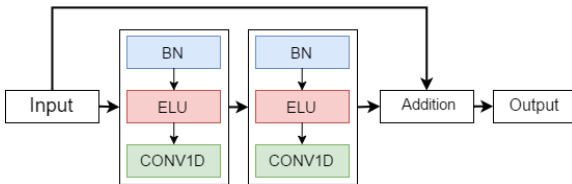
# Model

- Residual CNN
- 72 blocks, 2M parameters



**Figure 10:** Residual block

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



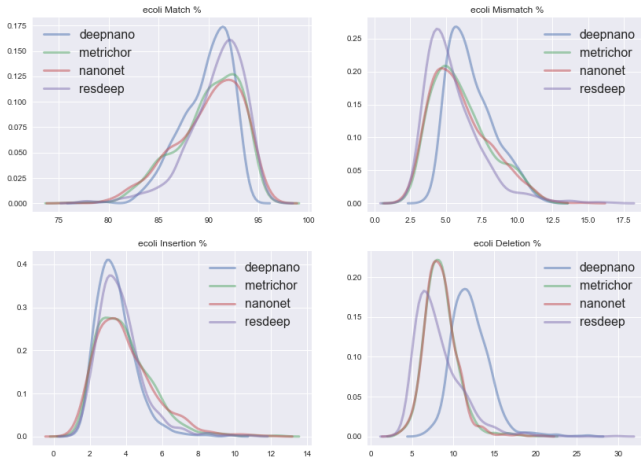
**Figure 10:** Residual block



## Results

---

# Results



**Figure 11:** KDE plot of alignment operations for E. Coli

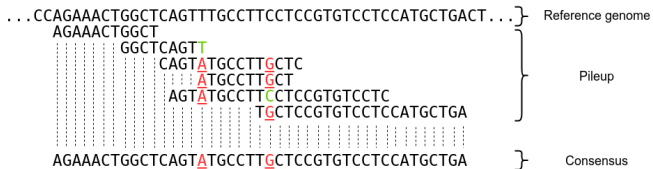
**Table 1:** Alignment specifications of E. Coli R9 basecalled reads

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

**Table 2:** Ecoli R9 basecalled read lengths in base pairs

	<b>median</b>	<b>mean</b>	<b>std</b>
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

## Consensus from pileup



**Figure 12:** Consensus from pileup

## 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	<b>99.2361</b>	<b>0.6474</b>	<b>0.1165</b>	<b>0.5510</b>

Consensus from de novo assembly obtained using assembler *ra*<sup>1</sup> and compared to the reference using *dnadiff* present in the Mummer<sup>2</sup>

---

<sup>1</sup><https://github.com/rvaser/ra>

<sup>2</sup><https://github.com/garviz/MUMmer>

Consensus from de novo assembly obtained using assembler *ra*<sup>1</sup> and compared to the reference using *dnadiff* present in the Mummer<sup>2</sup>

**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	<b>99.06</b>	98.47
Edit distance	60418	<b>46686</b>	74341

<sup>1</sup><https://github.com/rvaser/ra>

<sup>2</sup><https://github.com/garviz/MUMmer>



**Table 5:** Base calling speeds measured in *base pairs per second*<sup>3</sup>

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

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

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

- Scaled Exponential Linear Units (SELU), Jun 2017

- 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?**