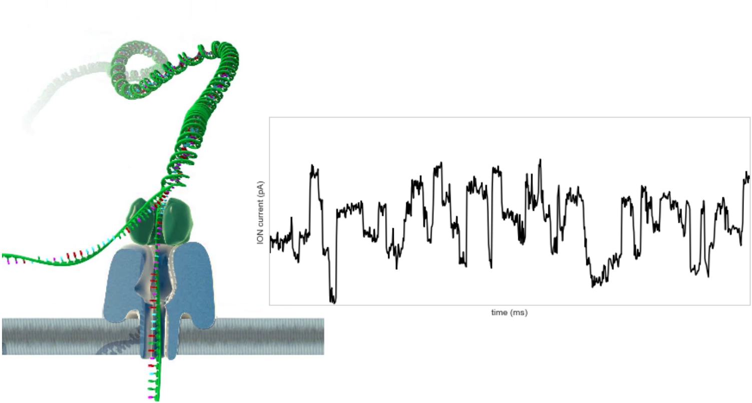
Deep Learning Model for Base Calling of MinION Nanopore Reads

Marko Ratković
Associate Profesor Mile Šikić, PhD

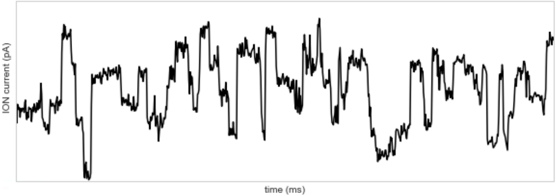
University of Zagreb
Faculty of Electrical Engineering and Computing

MinION

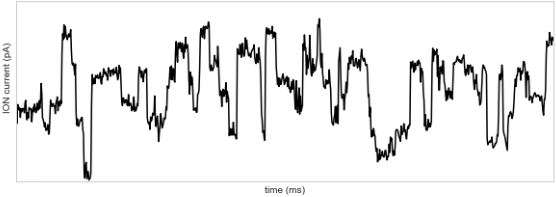
Technology



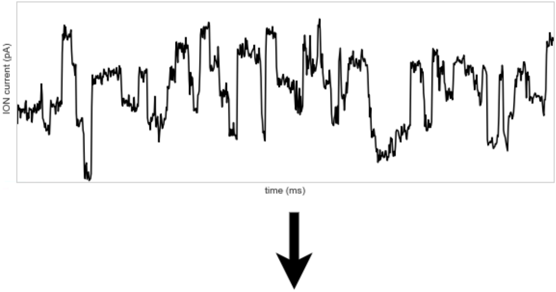
Basecalling



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

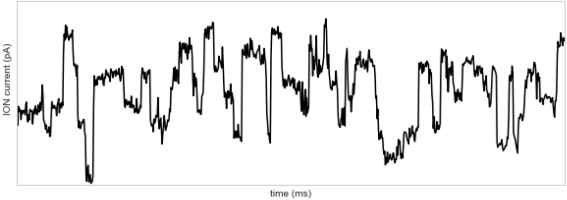
- local basecalling
- open-source
- speed, accuracy

Existing solutions

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

Idea?

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

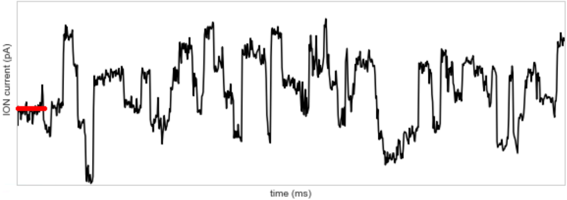


Existing solutions

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

Idea?

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

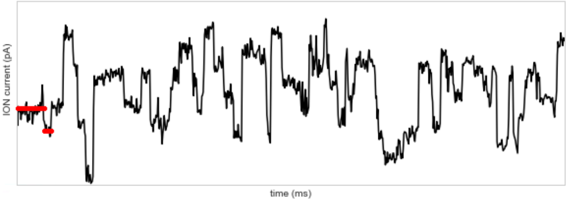


Existing solutions

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

Idea?

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

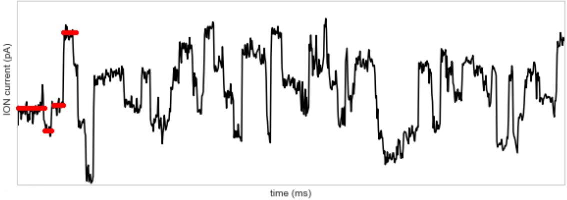


Existing solutions

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

Idea?

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

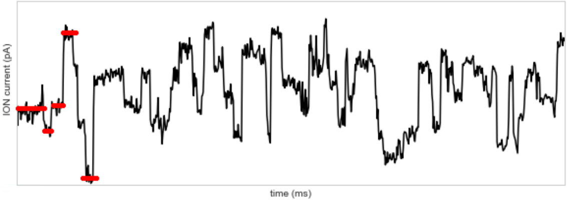


Existing solutions

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

Idea?

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

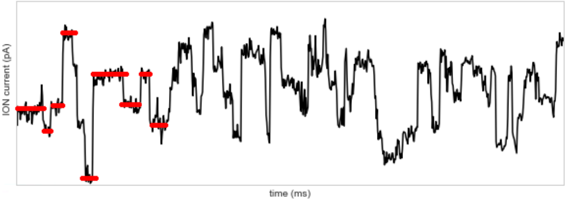


Existing solutions

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

Idea?

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

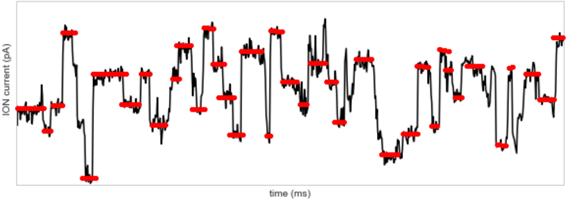


Existing solutions

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

Idea?

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



Proposed solution

end2end, CNN, CTC loss

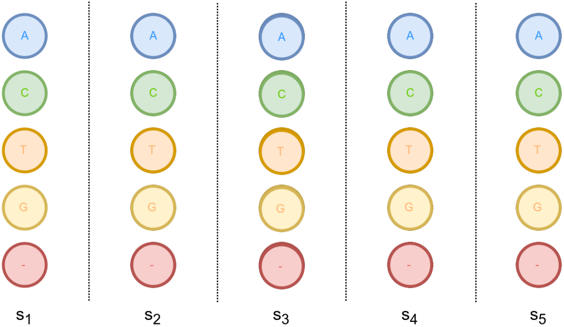
speed, paralelization, sequental, eliminate shit

variable length loss function

CTC loss

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

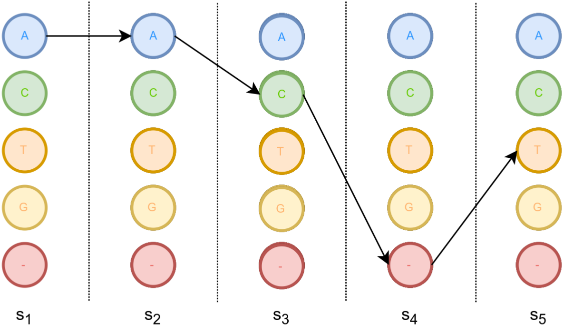
Figure 1: Path "AAC-T"



CTC loss

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

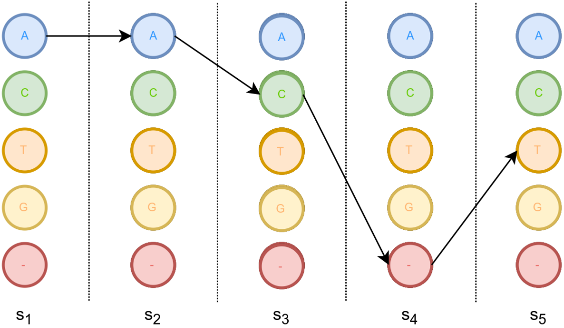
Figure 1: Path "AAC-T"



CTC loss

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

Figure 1: 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

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

CTC loss

Idea: decode sequence from fixed-width output

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

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

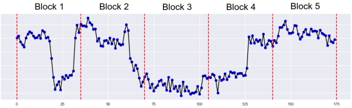
CTC loss

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:

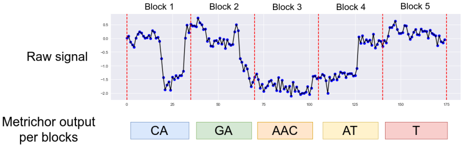
$$L(D) = -\sum_{(X,Y)\in D} lnP(Y|X) \tag{4}$$

Training

Raw signal



Training

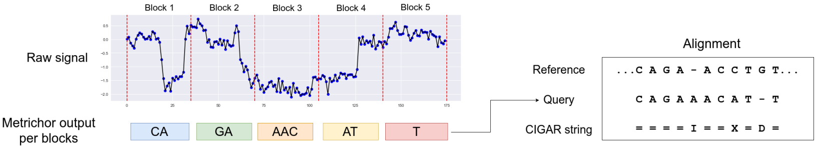


Raw signal

Metrichor output per blocks

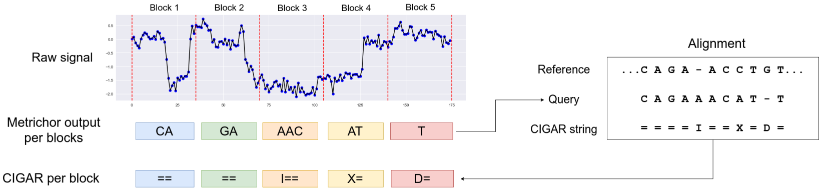
CA GA AAC AT T

Training



```
...C A G A - A C C T G T...
   C A G A A A C A T - T
   = = = = I = = X = D =
```

Training



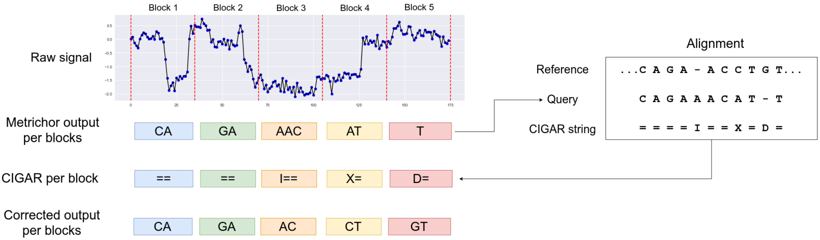
Raw signal

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 =

Training



Raw signal					
Metrichor output per blocks	CA	GA	AAC	AT	T
CIGAR per block	==	==	I==	X=	D=
Corrected output per blocks	CA	GA	AC	CT	GT

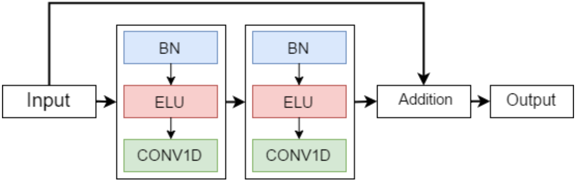
Alignment

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 =

Model

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

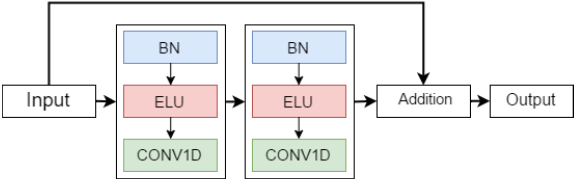
Figure 2: Residual block



Results

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

Figure 3: Residual block



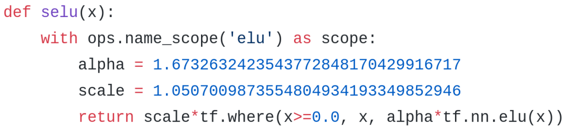
Future work

- Scaled Exponential Linear Units (SELU), Jun 2017

Future work

- Scaled Exponential Linear Units (SELU), Jun 2017

Figure 4: Implementation



```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))
```

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?