FZI Research Center for Information Technology

A PyTorch-Compatible Spike Encoding Framework for Energy-Efficient Neuromorphic Applications

Alexandru Vasilache (1, 2), Jona Scholz (1), Vincent Schilling (2), Sven Nitzsche (1, 2), Florian Kaelber (3), Johannes Korsch (3), Juergen Becker (2)

(1) FZI Research Center for Information Technology, Karlsruhe, Germany
(2) Karlsruhe Institute of Technology, Karlsruhe, Germany
(3) NXP Semiconductors Germany GmbH, Munich, Germany

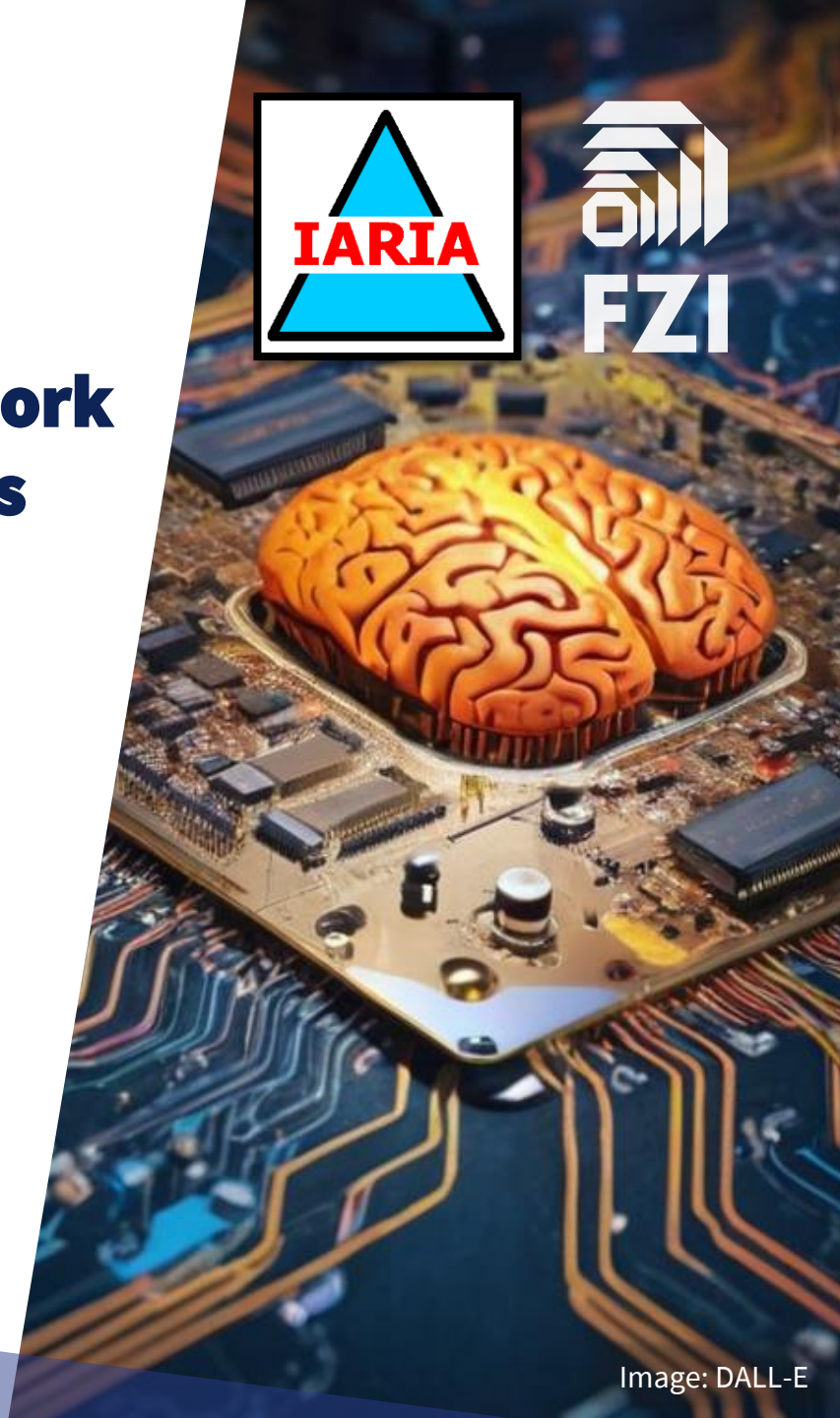


Image: DALL-E

16 May 2025

M.Sc. Alexandru Vasilache, FZI (vasilache@fzi.de)

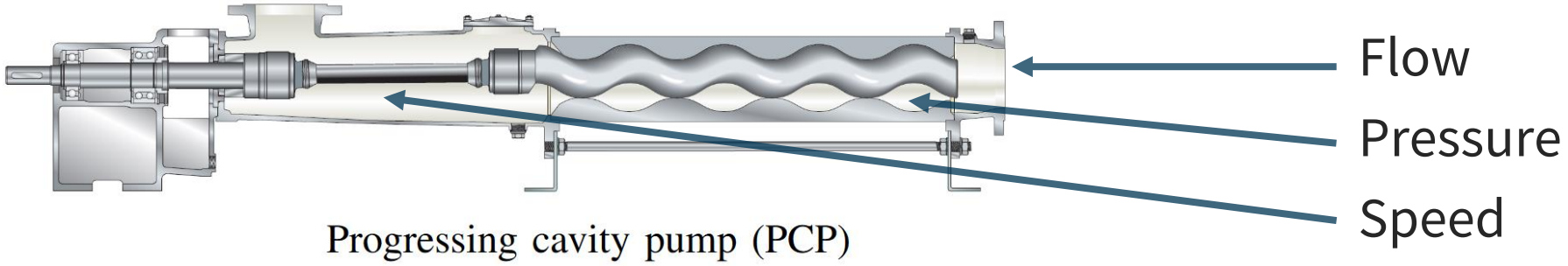
Alexandru Vasilache

Research Scientist - Forschungszentrum Informatik (FZI)

- Education:
  - B.Sc. & M.Sc. in Computer Science @ Karlsruhe Institute of Technology (KIT)
    - Specialization: AI & Robotics
  - Currently Ph.D. Student @ Karlsruhe Institute of Technology (KIT)
- Research Topics:
  - Neuromorphic Computing:
    - Spiking Neural Networks (SNNs)
    - Energy Efficiency in Embedded Neural Networks

Motivation

Spiking Neural Networks (SNNs) for Predictive Maintenance



Progressing cavity pump (PCP)

- Option 1:
  - Multiple Expensive Sensors

Image: Baoji

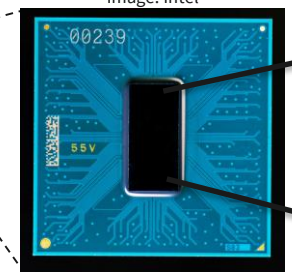
Image: nieruf

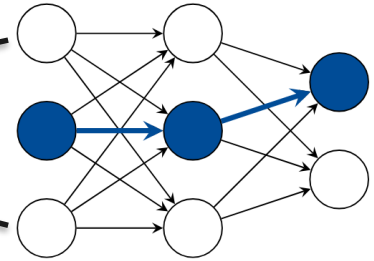
Image: Sensata-BEI

- Option 2:
  - Single Vibration Sensor
  - Embedded Spiking Neural Network (SNN)

Image: Siemens

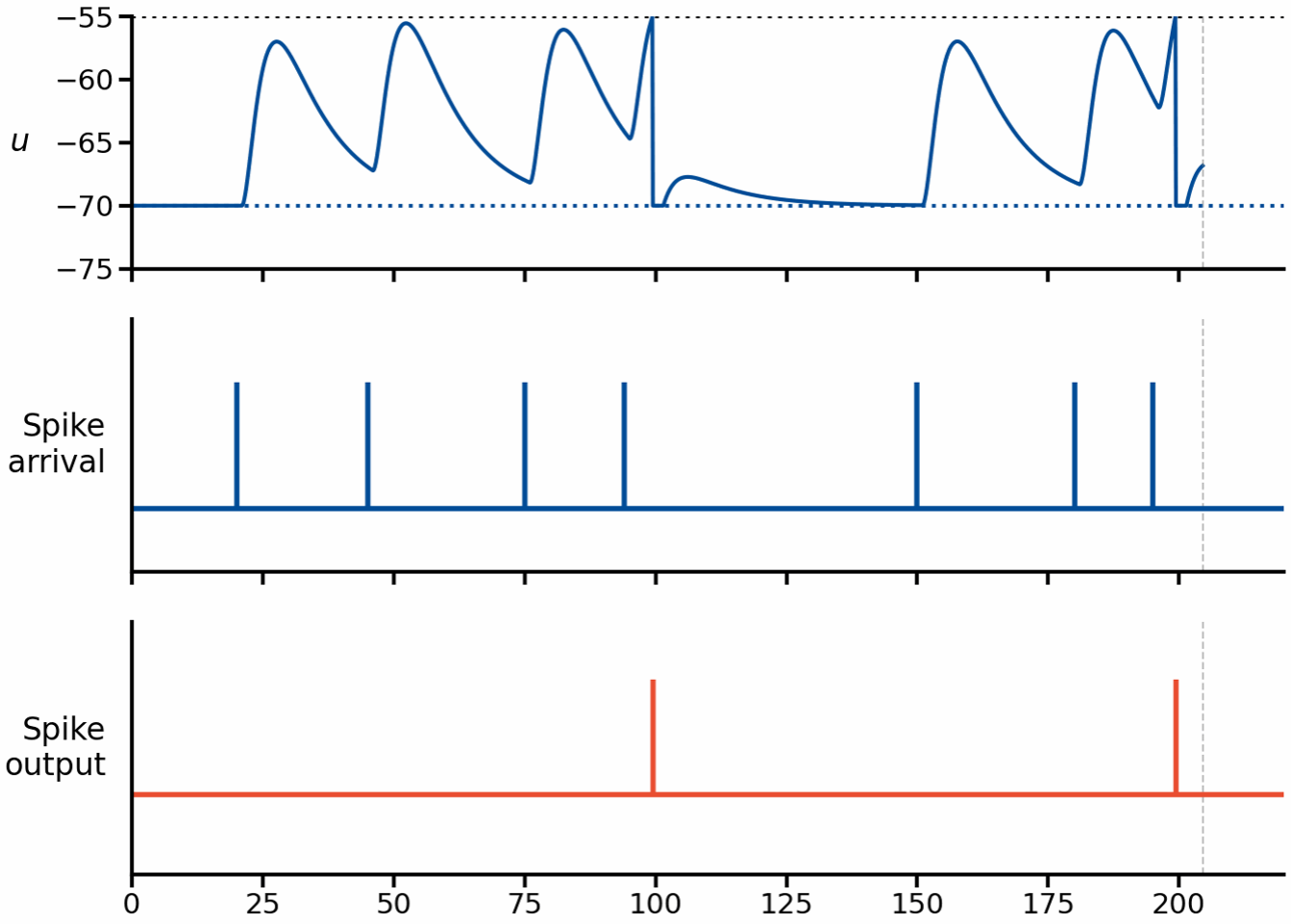
Image: Intel



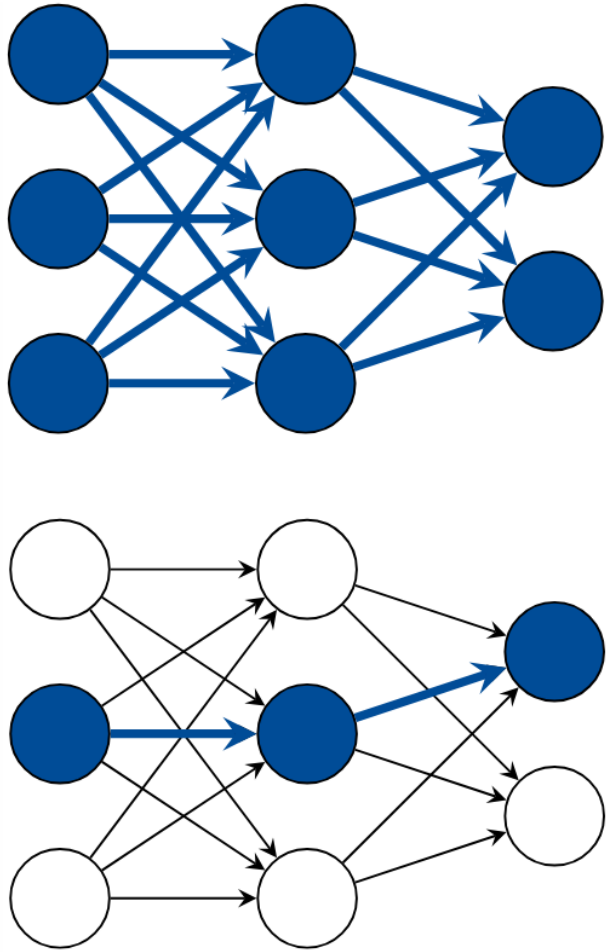


Spiking Neural Networks (SNNs)

Spiking Neurons



Sparsity

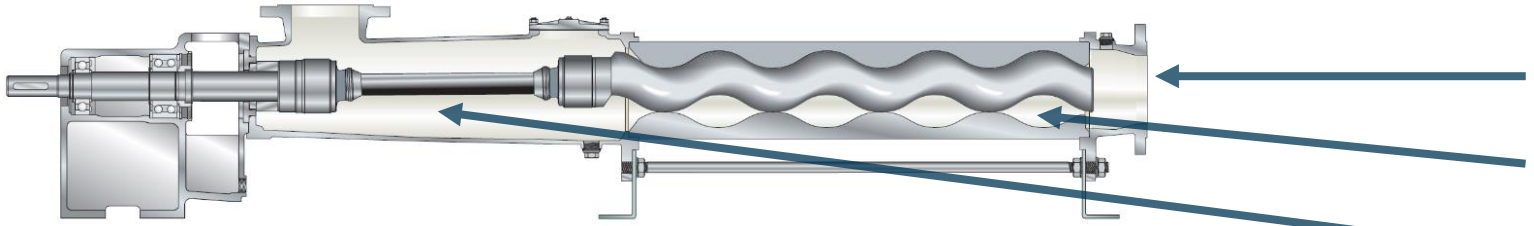


ANN

SNN

Motivation

Spiking Neural Networks (SNNs) for Predictive Maintenance



Progressing cavity pump (PCP)

Flow

Pressure

Speed

- Option 1:
  - Multiple Expensive Sensors

Image: Baoji

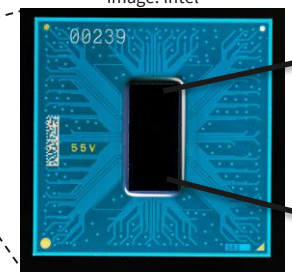
Image: nieruf

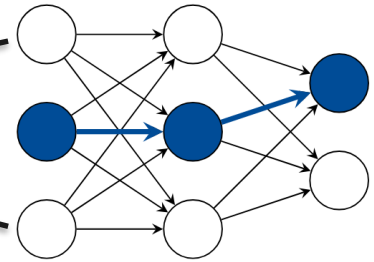
Image: Sensata-BEI

- Option 2:
  - Single Vibration Sensor – Numeric Values Spikes
  - Embedded Spiking Neural Network (SNN)

Image: Siemens

Image: Intel

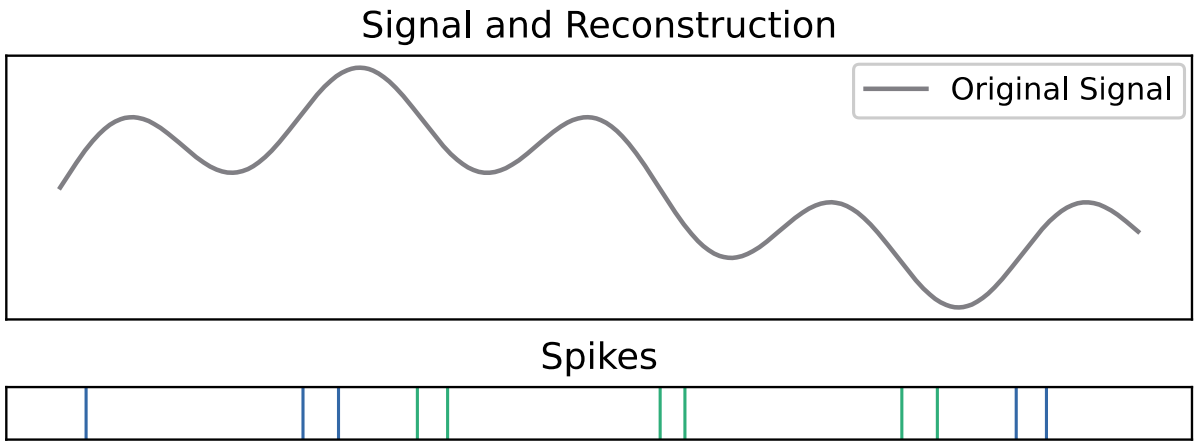




Spike Encoding

Encoding

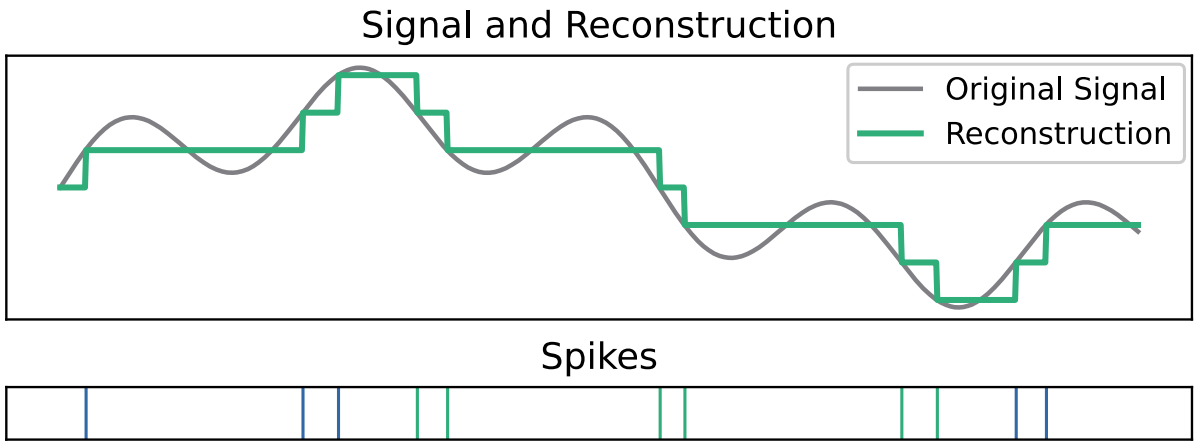
```
signal = tensor([0.0, 0.2, 0.4, 0.2, 0.0])

sf = StepForwardConverter(threshold=0.1)
spikes = sf.encode(signal)

# returns up/down spikes:
# up:   [0, 1, 1, 0, 0]
# down: [0, 0, 0, 0, 1]
```



Spike Encoding

Decoding

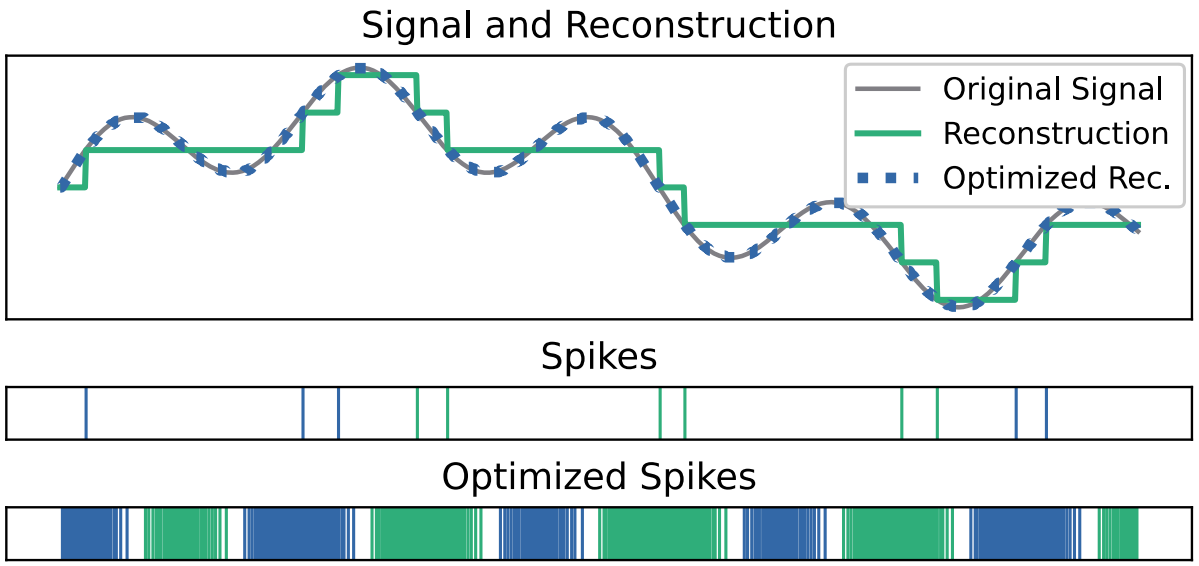
```
signal = tensor([0.0, 0.2, 0.4, 0.2, 0.0])

sf = StepForwardConverter(threshold=0.1)
spikes = sf.encode(signal)

sf.decode(spikes)
# returns [0.0, 0.1, 0.2, 0.2, 0.1]
```



Spike Encoding

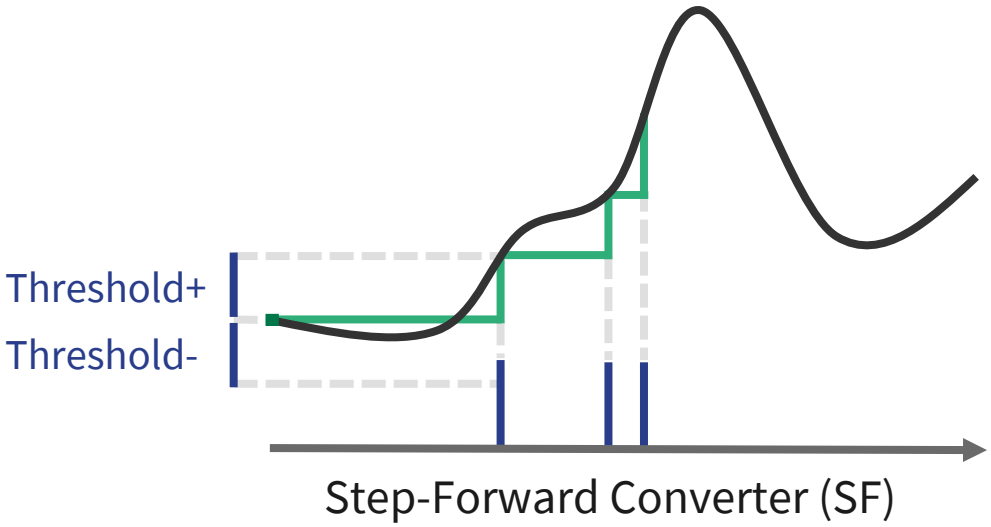
Optimization

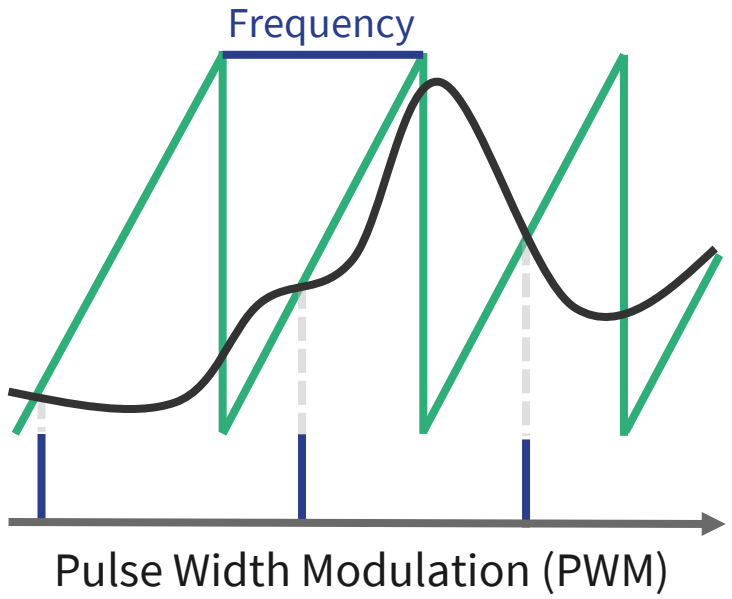
```
signal = tensor([0.0, 0.2, 0.4, 0.2, 0.0])

sf = StepForwardConverter(threshold=0.1)
spikes = sf.encode(signal)

sf.decode(spikes)

threshold = sf.optimize(signal)
```



Spike Encoding Algorithms

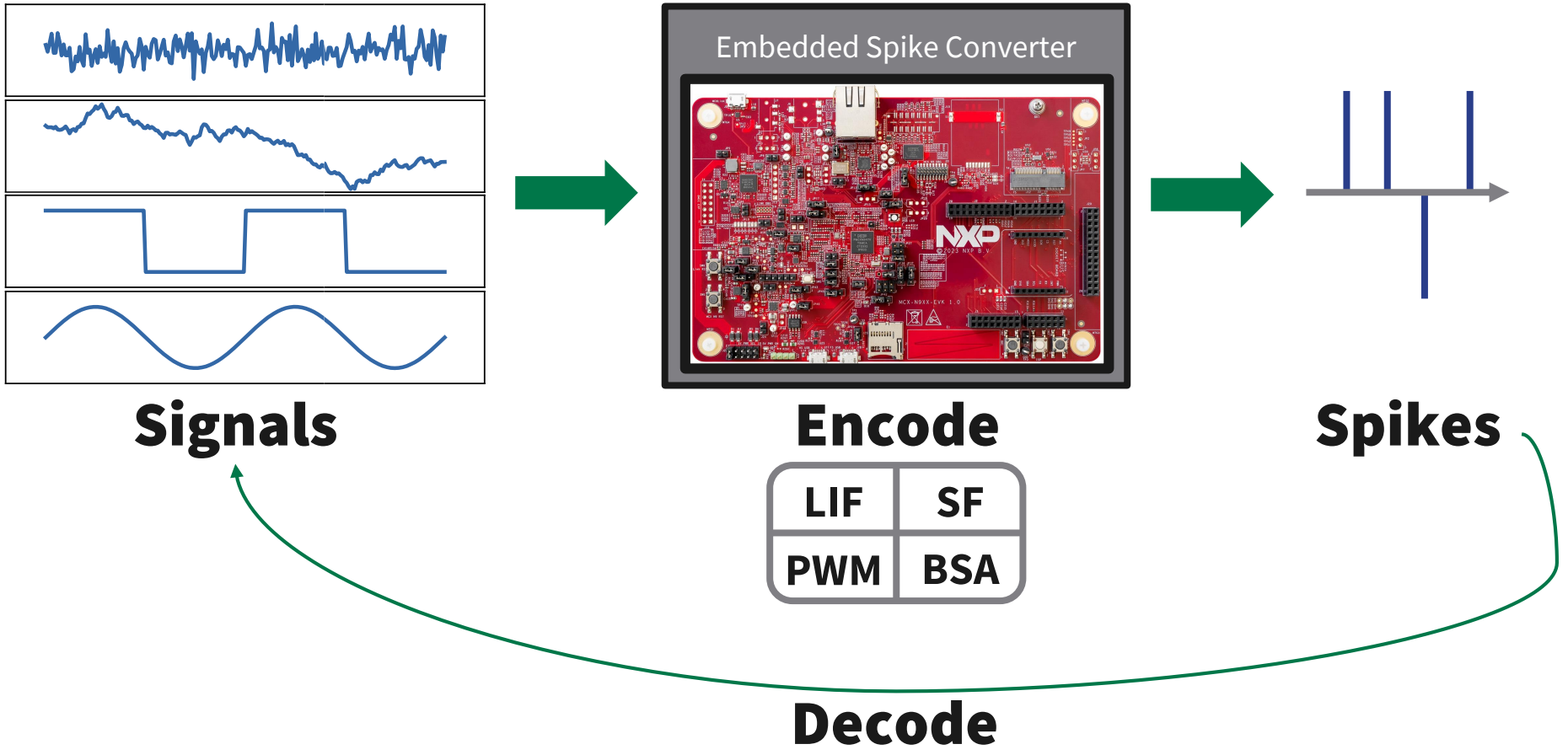
Step-Forward (SF) and Pulse Width Modulation (PWM)





Evaluation

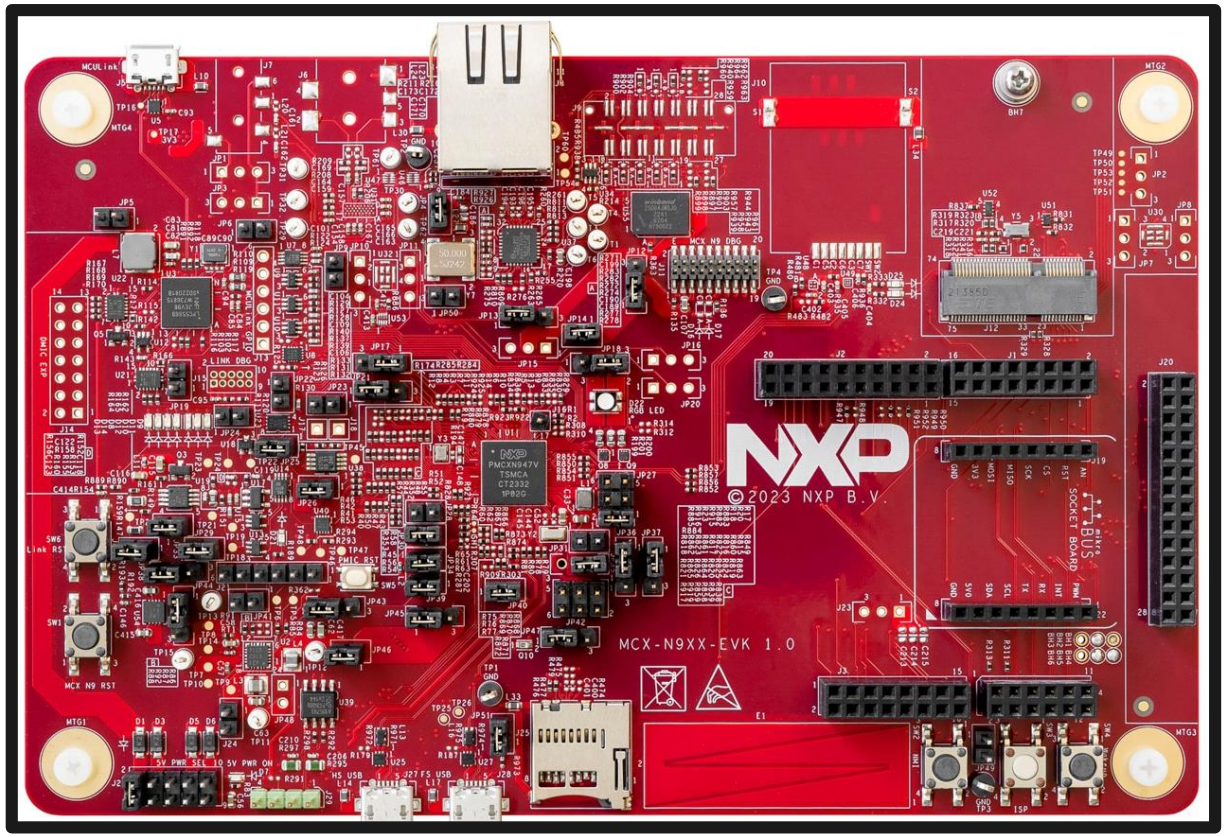
What is the best algorithm? Time Power Energy Sparsity Error



Embedded Platform

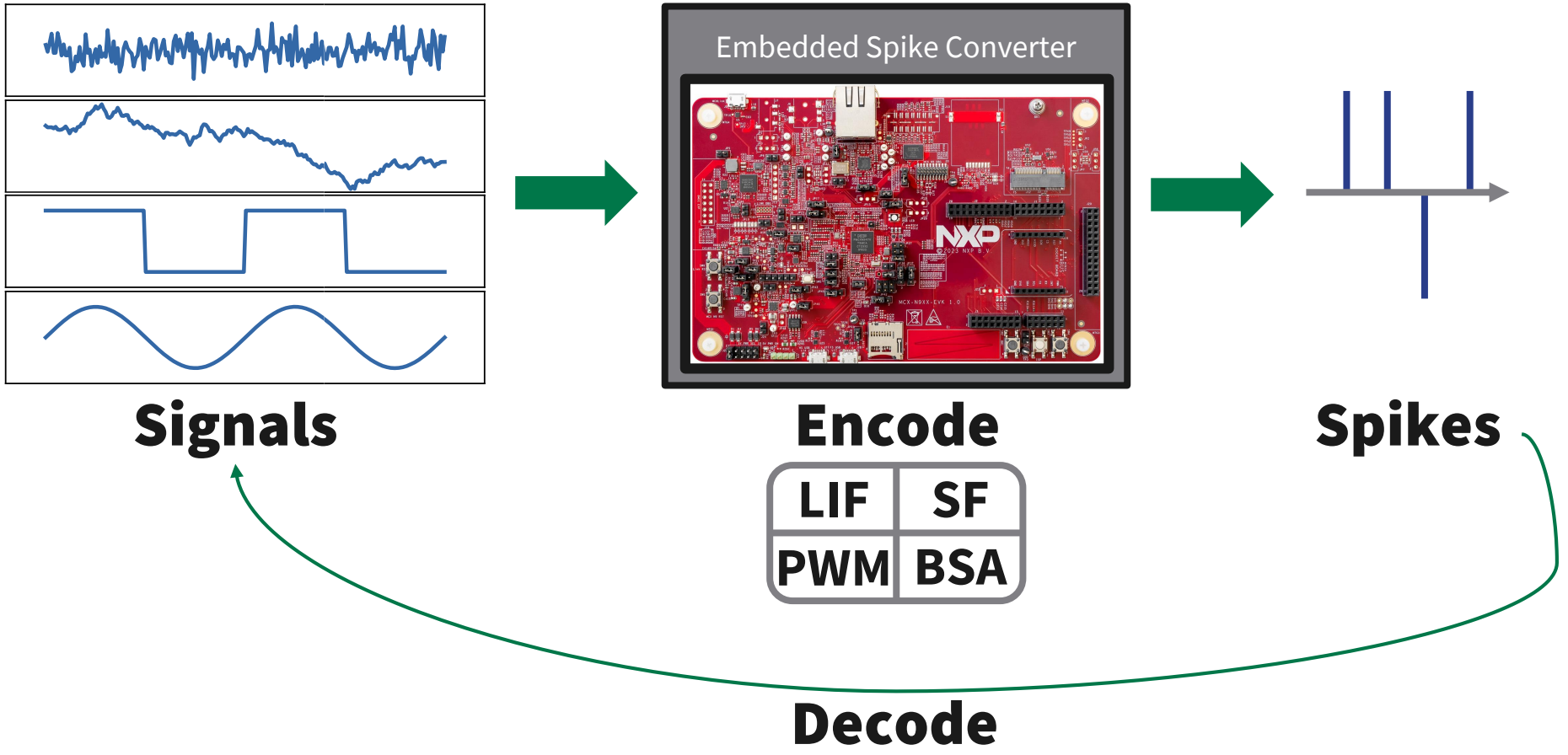
NXP MCX-N947

- Development board
- Efficient Power Measurement (on-board)
- On-board Debugger (MCU-Link)



Evaluation

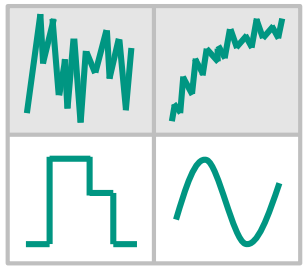
What is the best algorithm? Time Power Energy Sparsity Error

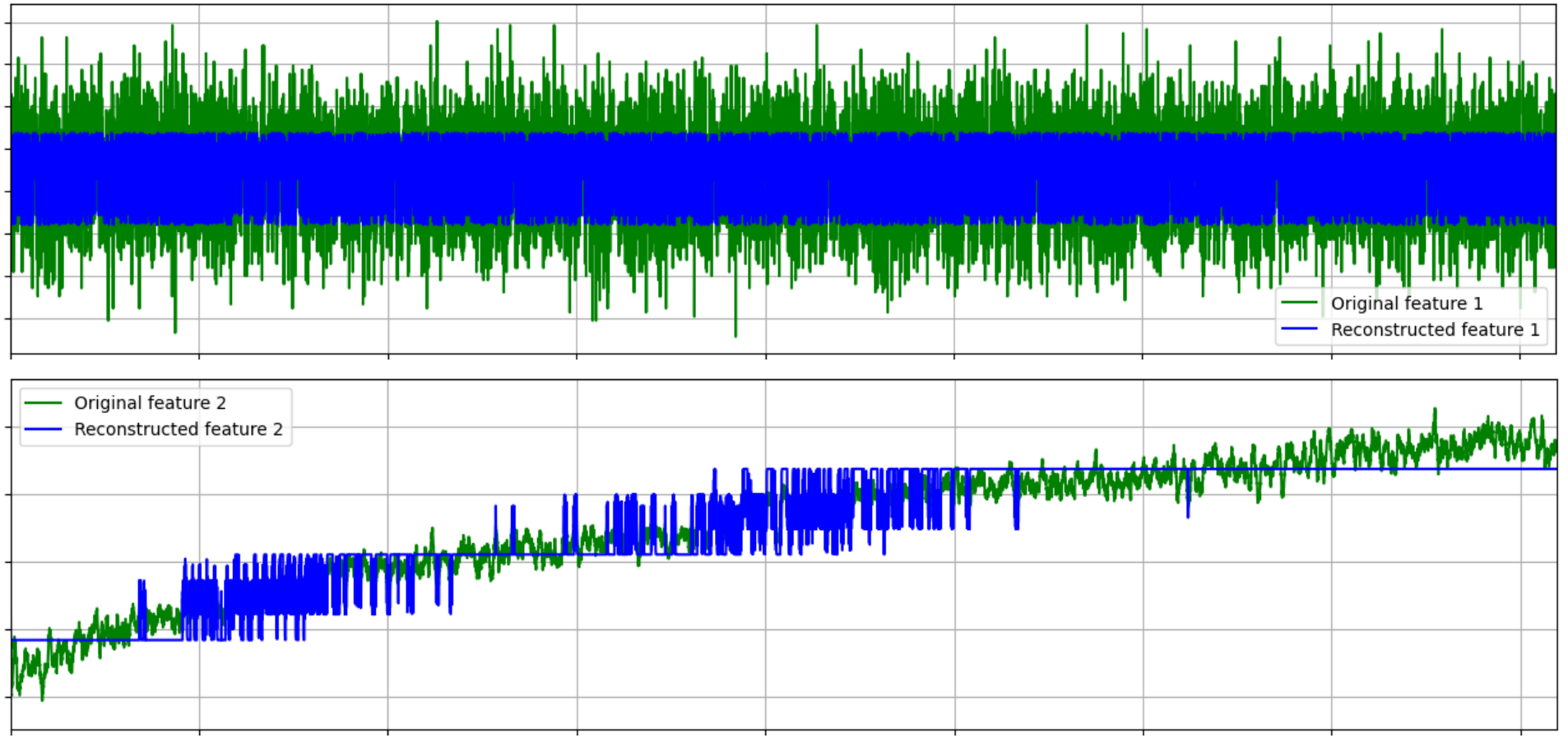


Evaluation: Leaky Integrate and Fire (LIF) Encoding

What is the best algorithm?

LIF	SF
PWM	BSA

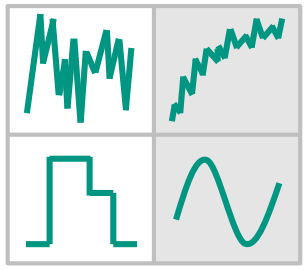


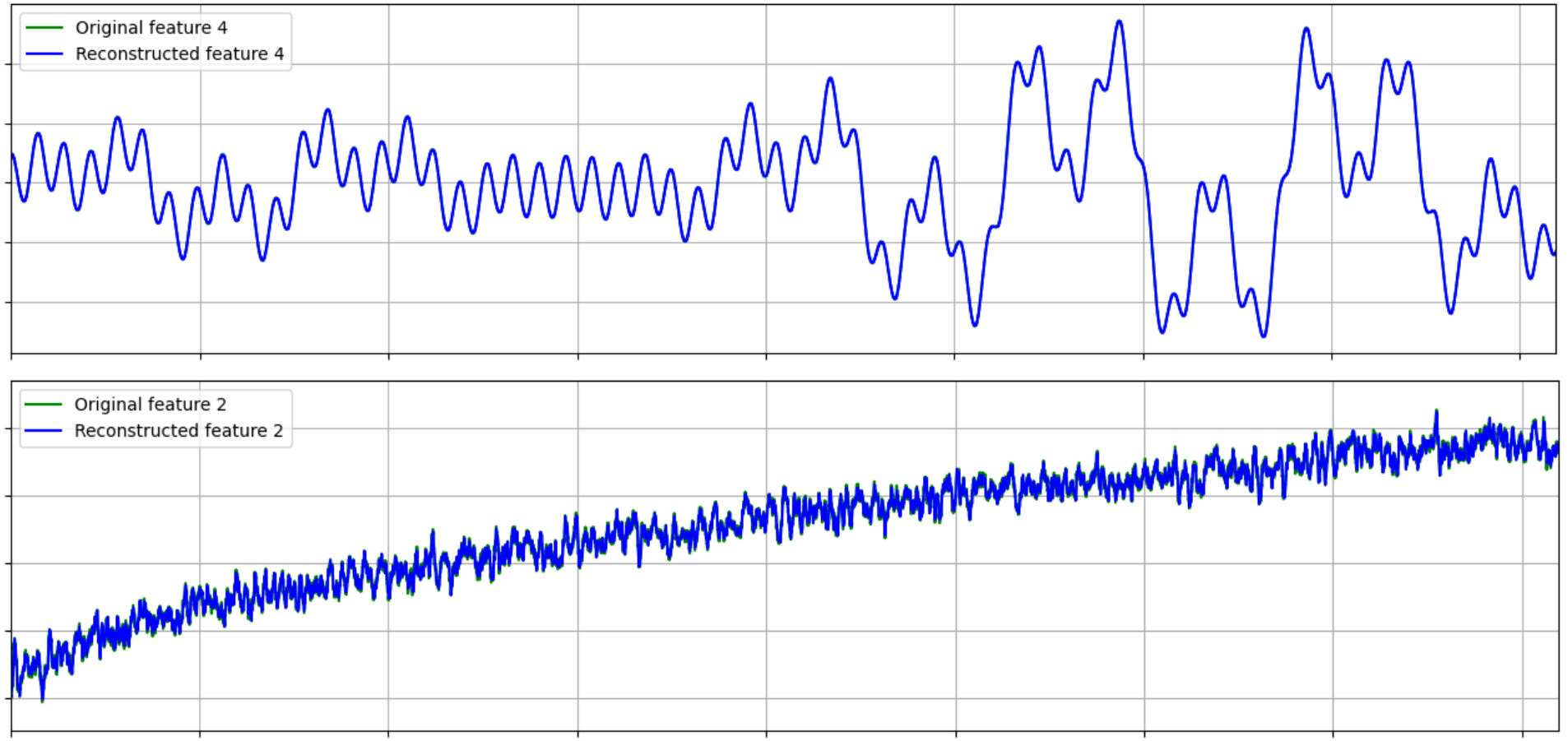


Evaluation: Step-forward encoding (SF)

What is the best algorithm?

LIF	SF
PWM	BSA

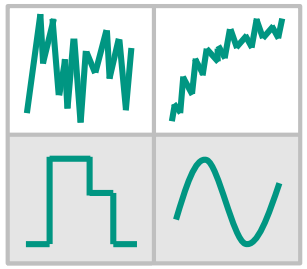


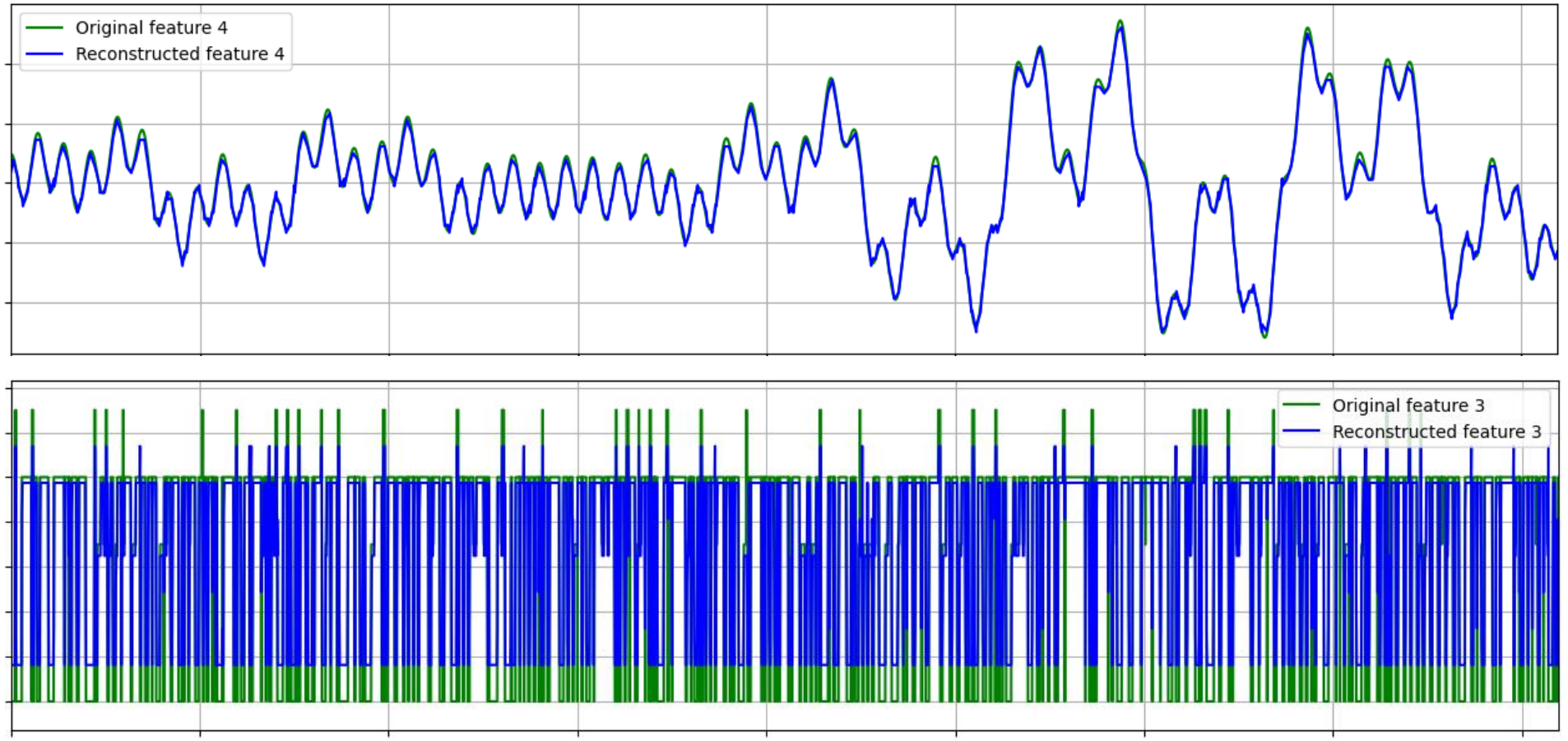


Evaluation: Pulse-width modulation (PWM)

What is the best algorithm?

LIF	SF
PWM	BSA

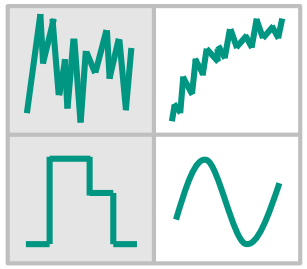


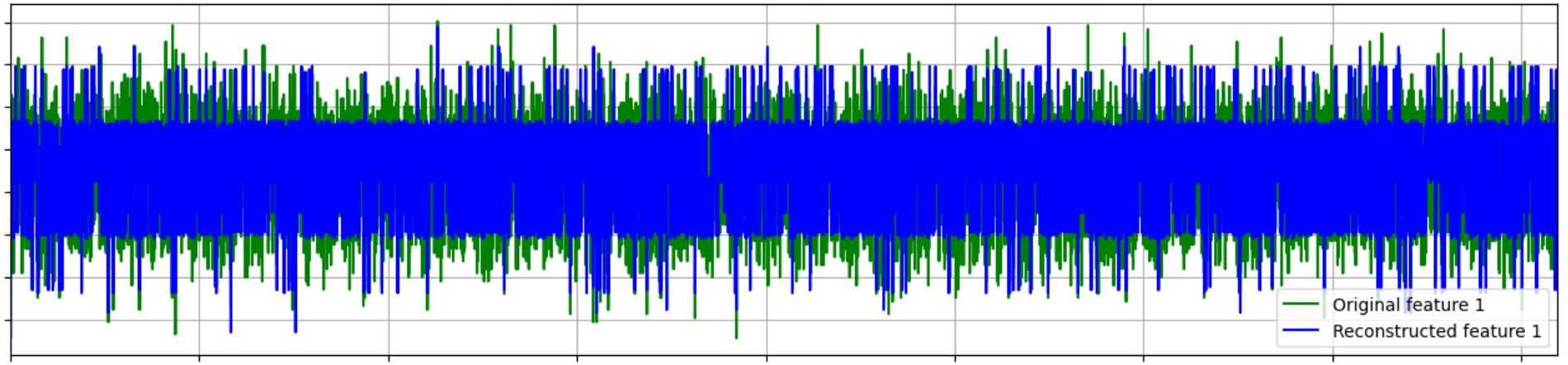


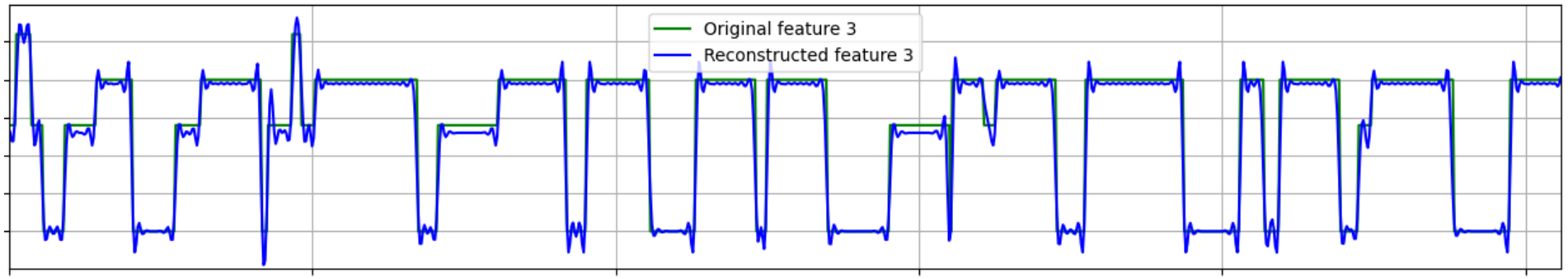
Evaluation: Ben's spiker algorithm (BSA)

What is the best algorithm?

LIF	SF
PWM	BSA

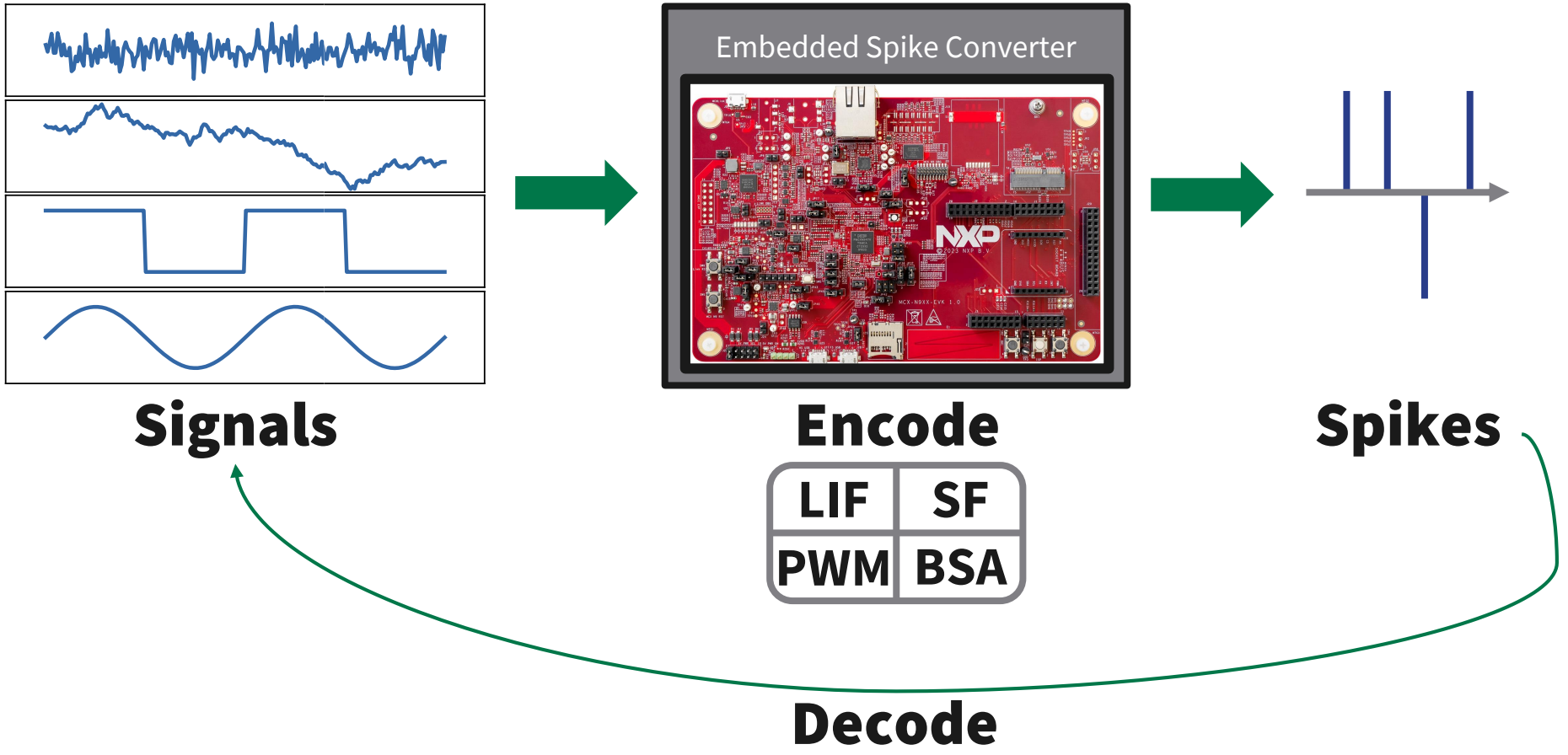






Evaluation

What is the best algorithm? Time Power Energy Sparsity Error



Evaluation

What is the best algorithm?

- Best Error: **SF** — Worst: **PWM**
- Best Sparsity: **PWM** — Worst: **BSA**
- Best Time: **SF** — Worst: **BSA**
- Best Power: **SF** — Worst: **BSA**
- Best Energy: **SF** — Worst: **BSA**

Encoding Method	LIF	SF	PWM	BSA
Error (MSE)	0.17	**0.16**	0.32	0.23
Sparsity (%)	51.9	30.6	**17.7**	53.1
Time (s)	0.13	**0.12**	0.69	2.24
Power (nW)	9.71	**6.87**	11.4	20.0
Energy (pWh)	0.34	**0.24**	2.19	12.4

Spike Encoding Repository

github.com/Alex-Vasilache/Spike-Encoding

Encoding

```
signal = tensor([0.0, 0.2, 0.4, 0.2, 0.0])

sf = StepForwardConverter(threshold=0.1)
spikes = sf.encode(signal)
```

Decoding

```
sf.decode(spikes)
```

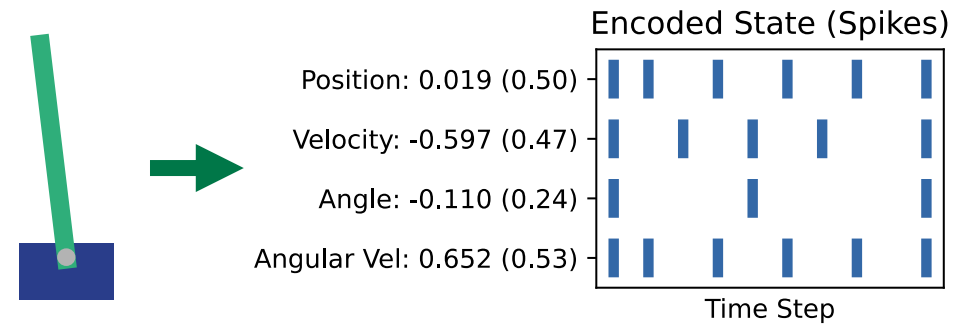
Optimization (minimize reconstruction error)

```
threshold = sf.optimize(signal)
```

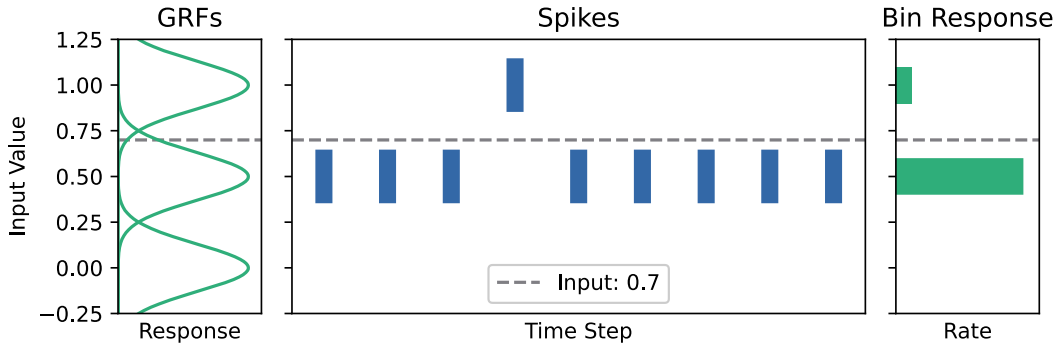
Supported Algorithms

- LIF-based encoding (LIF)
- Step-forward encoding (SF)
- Pulse-width modulation (PWM)
- Ben's spiker algorithm (BSA)
- Gymnasium encoder
- Bin encoder

Gymnasium Encoder

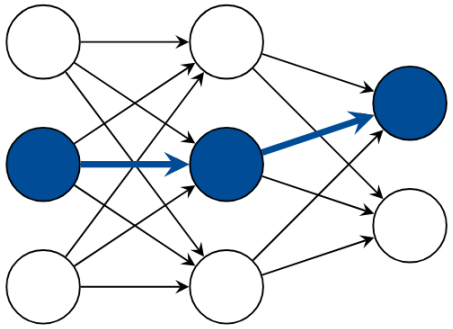


Bin Encoder

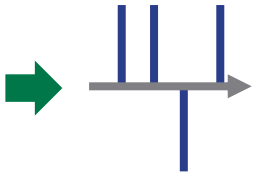


Conclusion

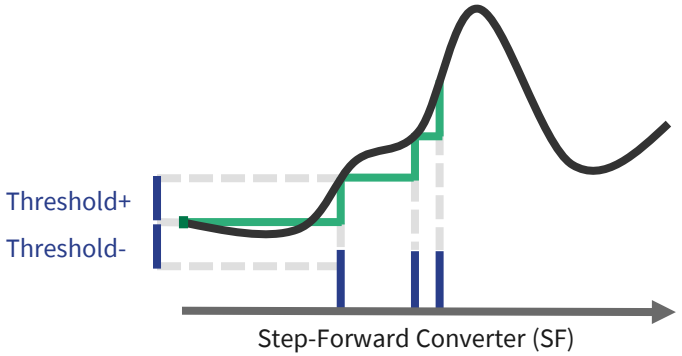
Spike Encoding Framework



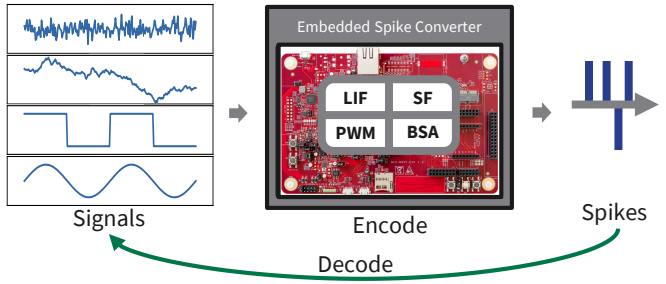
SNN



Spikes



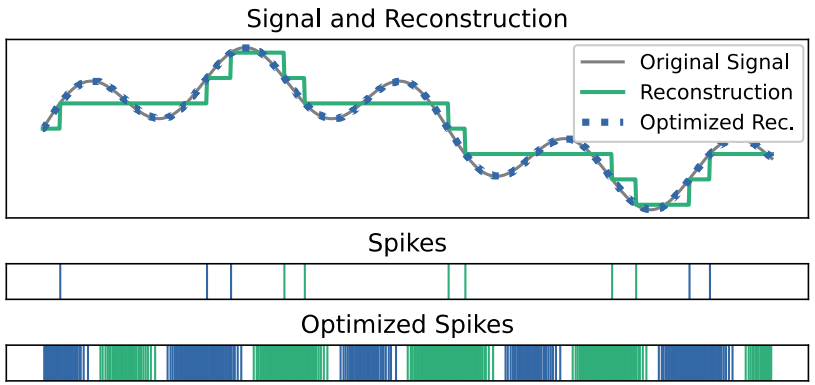
Encoders



Embedded Evaluation

Encoding Method	LIF	SF	PWM	BSA
Error (MSE)	0.17	**0.16**	0.32	0.23
Sparsity (%)	51.9	30.6	**17.7**	53.1
Time (s)	0.13	**0.12**	0.69	2.24
Power (nW)	9.71	**6.87**	11.4	20.0
Energy (pWh)	0.34	**0.24**	2.19	12.4

Results



Supported Algorithms

- LIF-based encoding (LIF)
- Step-forward encoding (SF)
- Pulse-width modulation (PWM)
- Ben's spiker algorithm (BSA)
- Gymnasium encoder
- Bin encoder

Repository

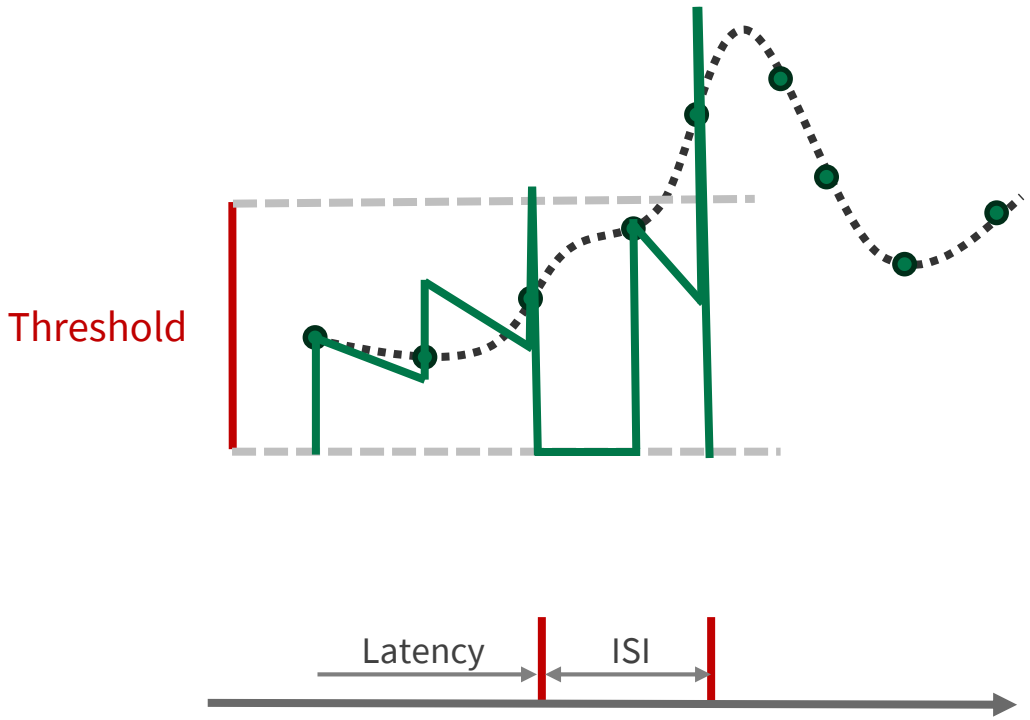
Thank You!

M.Sc. Alexandru Vasilache
Research Scientist @FZI
Embedded Systems and Sensors Engineering (ESS)

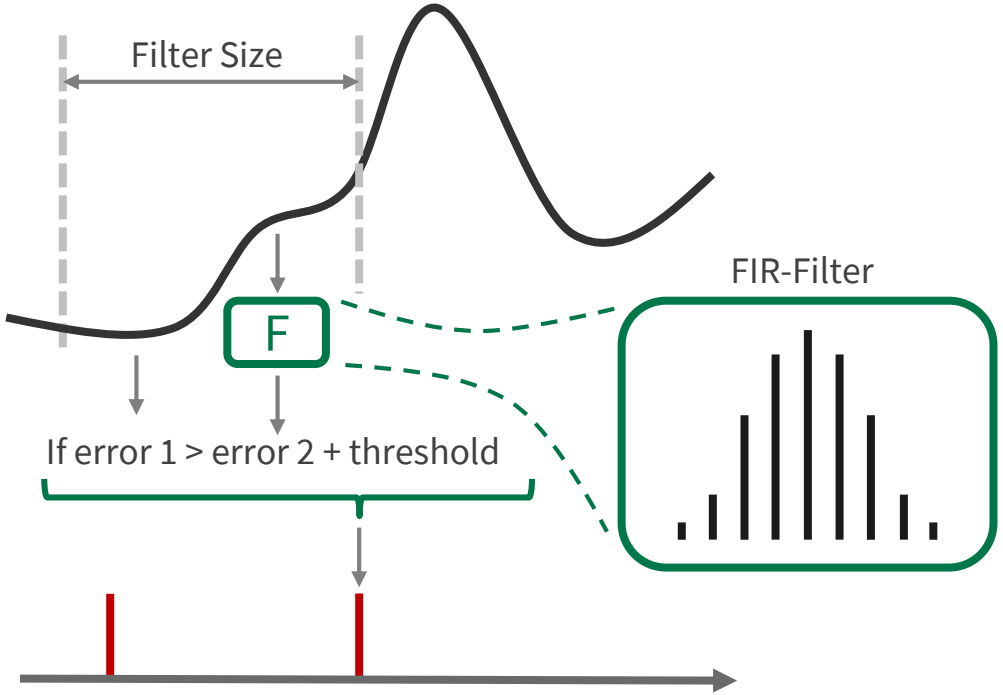
FZI Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
D-76131 Karlsruhe
+49 721 9654-180
vasilache@fzi.de

Spike Encoding Algorithms

Leaky Integrate and Fire encoding (LIF) and Bens Spiker Algorithm (BSA)



Leaky Integrate and Fire (LIF)



Ben's spiker algorithm (BSA)