Sleep Spindles as a Driver of Low Latency, Low Power ML in HLS4ML & TinyML

Hardware Development: ***Xiaohan Liu(Speaker)***, Jeffery Xu, Aidan Yokuda, Scott Hauck, Shih-Chieh Hsu
Neural Interfaces: Leo Scholl, Michael Nolan, Amy Orsborn
Neural Processing Algorithms: Trung Le, Eli Shlizerman

Neural Data – Sleep Spindles

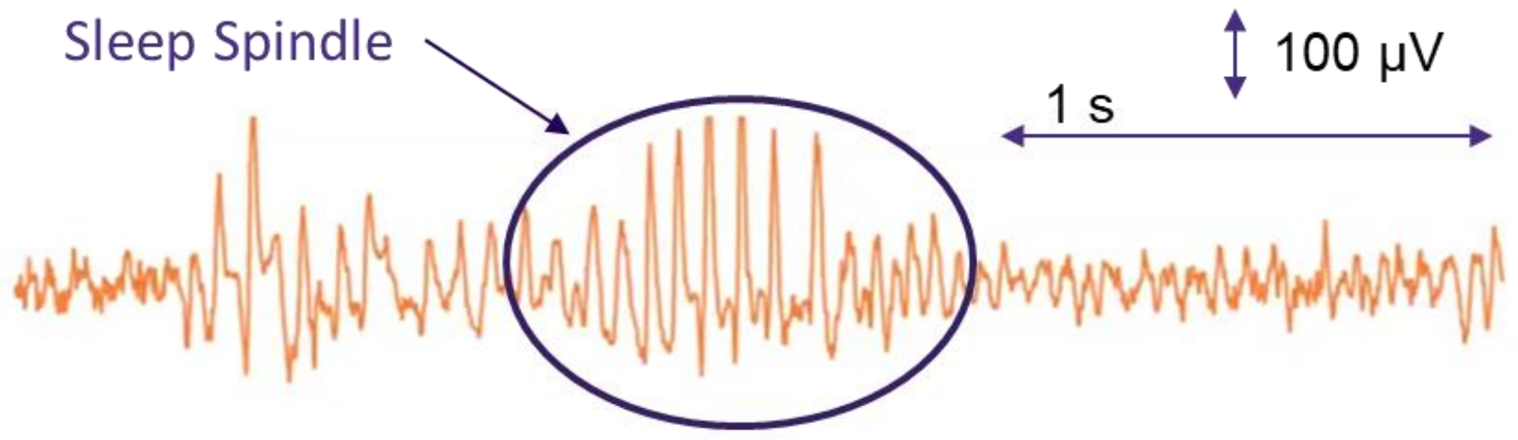


Fig. 1. Brain signal – Sleep Spindles[1]

Sleep Spindles Introduction [1]

- Rare low-frequency brain signals
- Primarily occur during sleep or rest
- Are believed to contribute to learning
- Lack of mechanistic understanding

Our goal

- Design and build a system that can help neuroscientists to understand the mechanism behind the theory

The Proposed System

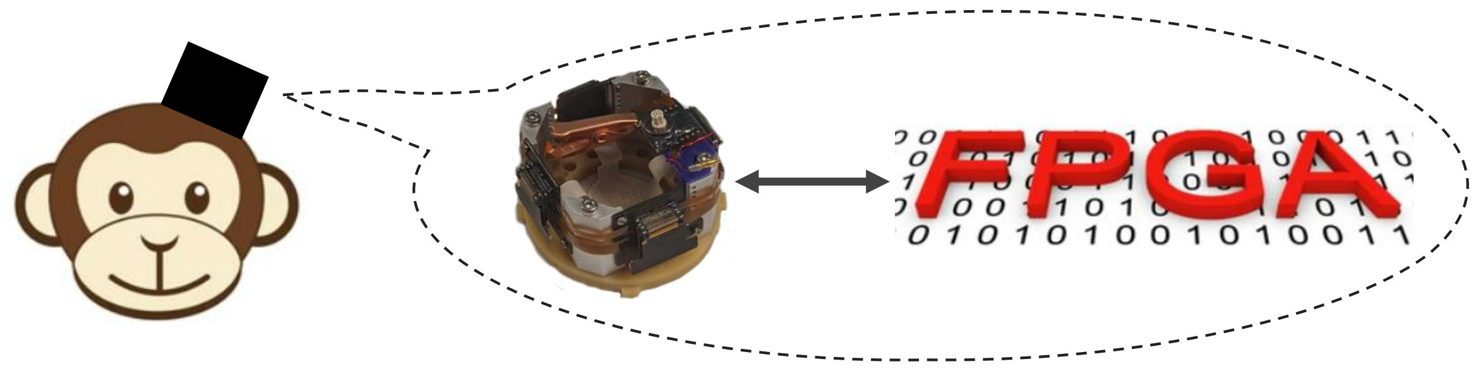


Fig. 2. Head-Mounted Device on Subject[2][3]

Head-Mounted Device components

- Headstage: Records brain signals from the subject
- Programmed FPGA: Processes brain signals and interacts with sleep spindles

Methods (HLS4ML & TinyML)

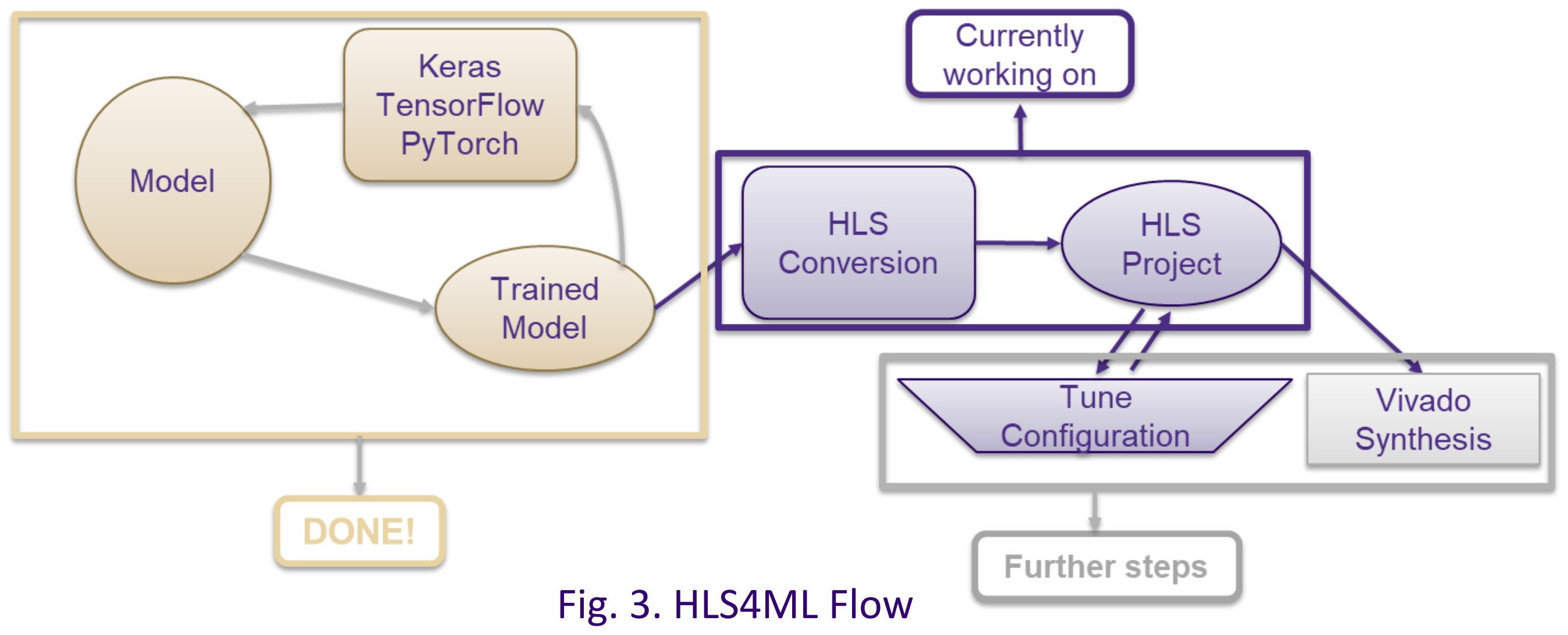


Fig. 3. HLS4ML Flow

The brain signals will be analyzed by a deep learning model, which will be pushed through the HSL4ML.

TinyML will help us to deploy the model on an ultra low power FPGA.

Baseline Deep Learning Model

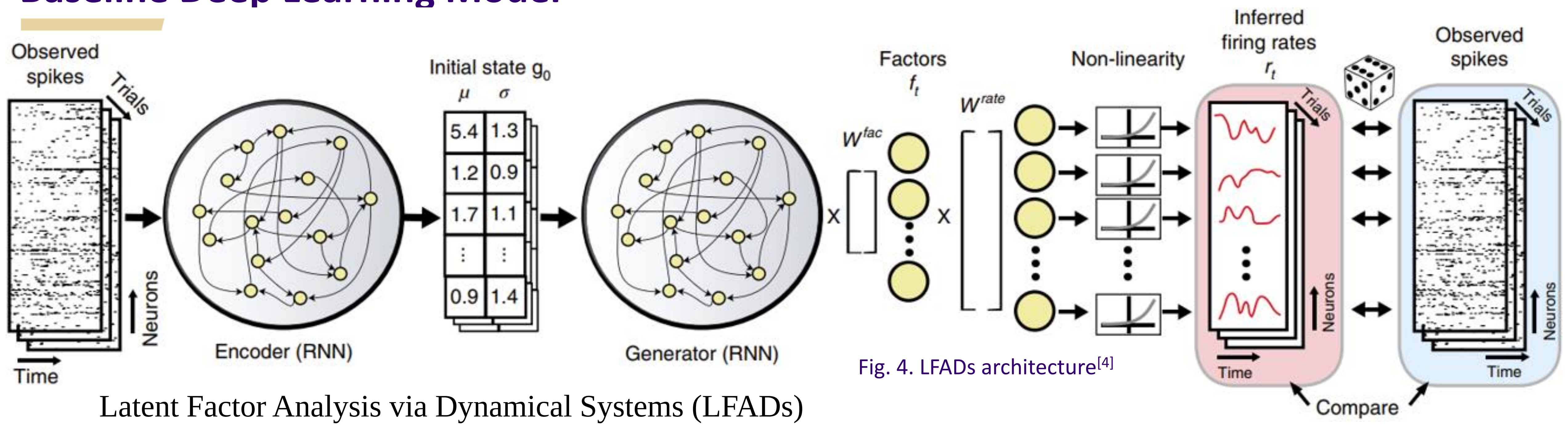


Fig. 4. LFADs architecture[4]

Latent Factor Analysis via Dynamical Systems (LFADs)

- RNN variational autoencoder (VAE) in tf.keras API
- Input: Neural spiking data
- Output: Firing Rates & LFADs Latent Factors

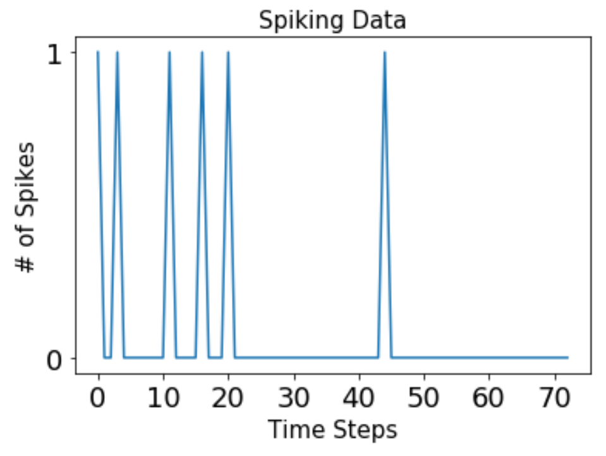


Fig. 5. Neural Spiking Data (Input)

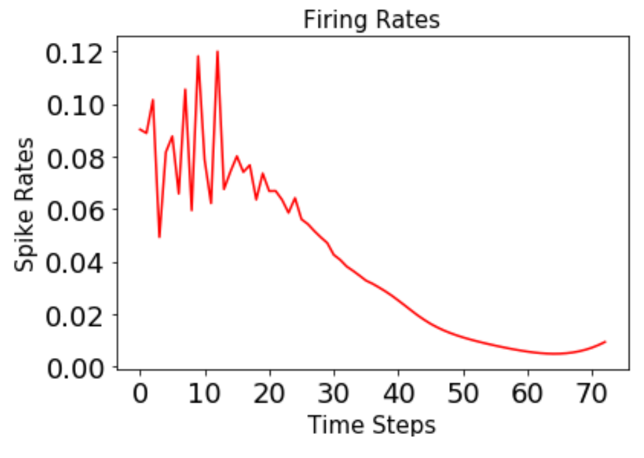
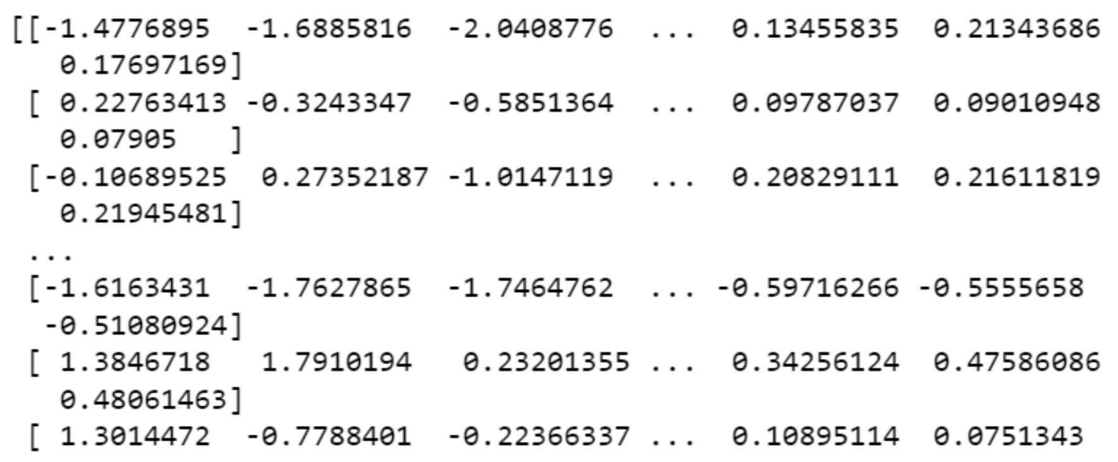


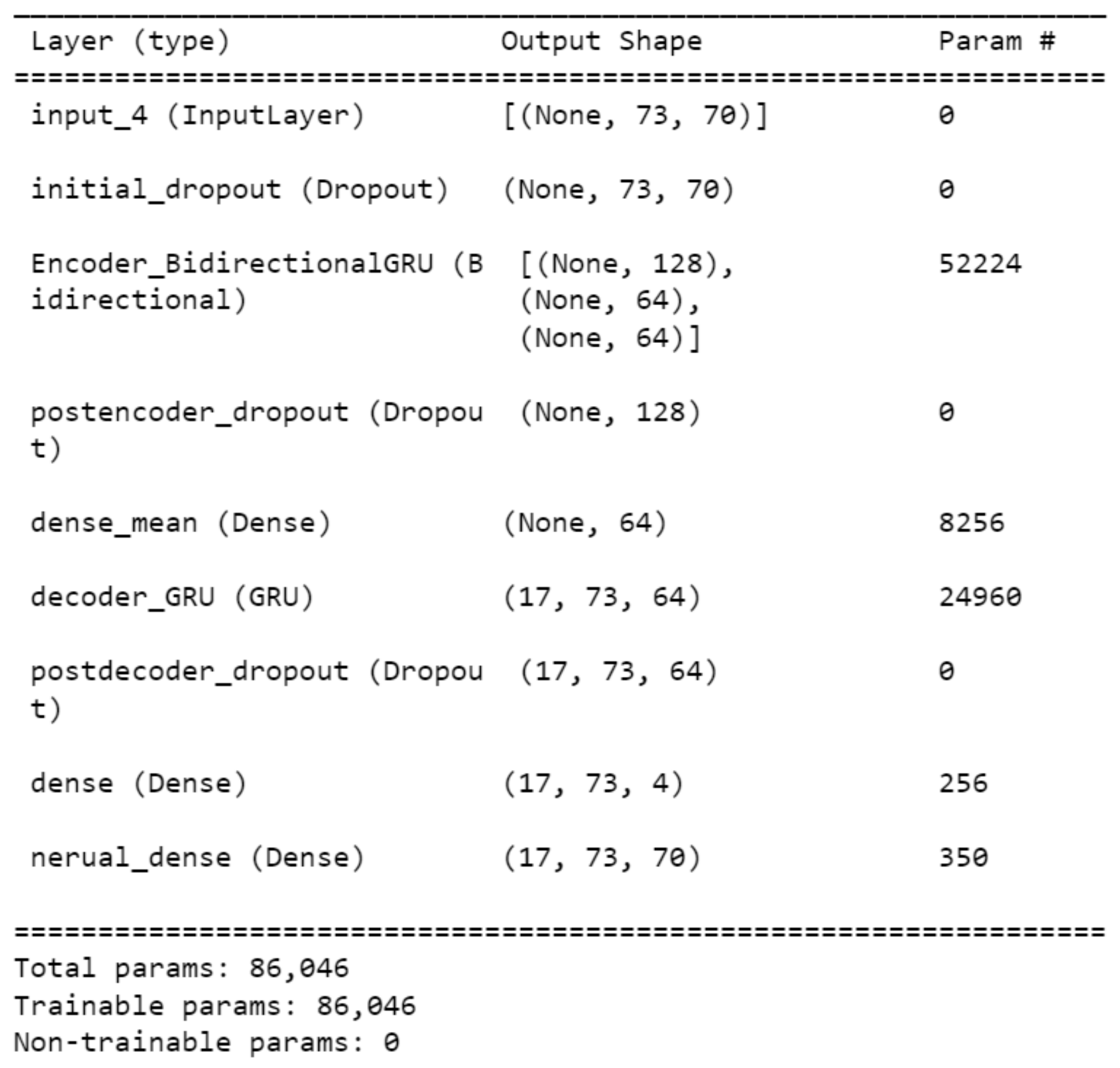
Fig. 6. Firing Rates (Output)



```
[[-1.4776895  -1.6885816  -2.0408776  ...  0.13455835  0.21343686
   0.17697169]
 [ 0.22763413 -0.3243347  -0.5851364  ...  0.09787037  0.09010948
   0.07905   ]
 [-0.10689525  0.27352187 -1.0147119  ...  0.20829111  0.21611819
   0.21945481]
 ...
 [-1.6163431  -1.7627865  -1.7464762  ... -0.59716266 -0.5555658
  -0.51080924]
 [ 1.3846718   1.7910194   0.23201355 ...  0.34256124  0.47586086
   0.48061463]
 [ 1.3014472  -0.7788401  -0.22366337 ...  0.10895114  0.0751343
```

Fig. 7. LFADs Latent Factors (Output)

Modified LFADs architecture



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_4 (InputLayer)        [(None, 73, 70)]          0

 initial_dropout (Dropout)   (None, 73, 70)            0

 Encoder_BidirectionalGRU (B  [(None, 128),            52224
 idirectional)                (None, 64),
                              (None, 64)]

 postencoder_dropout (Dropou  (None, 128)              0
 t)

 dense_mean (Dense)          (None, 64)                8256

 decoder_GRU (GRU)           (17, 73, 64)              24960

 postdecoder_dropout (Dropou  (17, 73, 64)             0
 t)

 dense (Dense)               (17, 73, 4)               256

 nerual_dense (Dense)        (17, 73, 70)              350

=================================================================
Total params: 86,046
Trainable params: 86,046
Non-trainable params: 0
```

Fig. 8. Modified LFADs Model Summary

By removing the gaussian sampling layer, LFADs are converted to an autoencoder, which is easier to be pushed through HLS4ML flow.

Performance Comparison per Trial

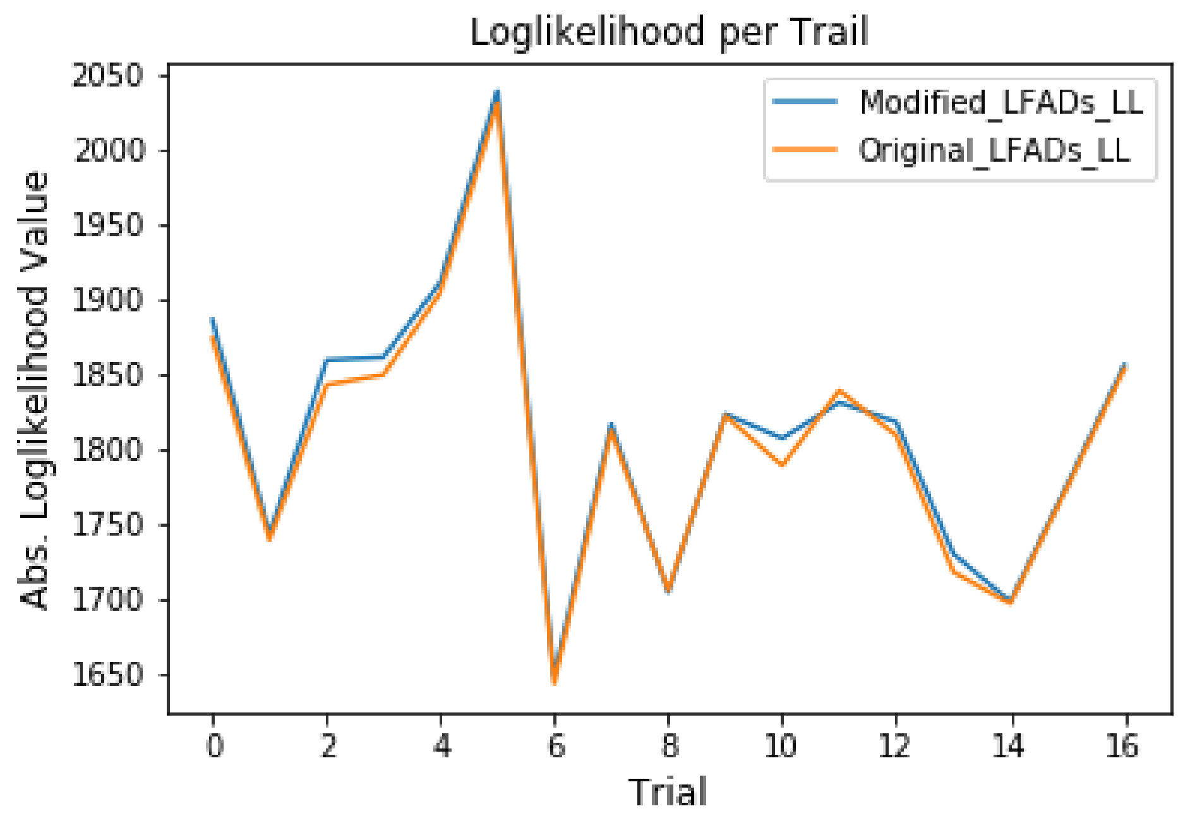


Fig. 9. Modified & Original LFADs Performance comparison

The negative log-likelihood is the evaluation metric of the LFADs. Minimized negative log-likelihood indicates an optimal performance.

For the same testing dataset, the numerical value of the negative log-likelihood from the modified LFADs matches to the original LFADs, which indicates that removing the gaussian sampling from LFADs is acceptable.

Challenges & Plans

Will need to add the gaussian sampling layer back to enhance the robustness of the model. This can be divided into 3 phases.

- Phase I: Analyze gaussian sampling layer's FPGA resource utilization and compare with an autoencoder (Done!)
- Phase II: Implement gaussian sampling layer in HLS4ML (Jeffery's current project)
- Phase III: Integrate the gaussian sampling layer with the LFADs autoencoder (Future)

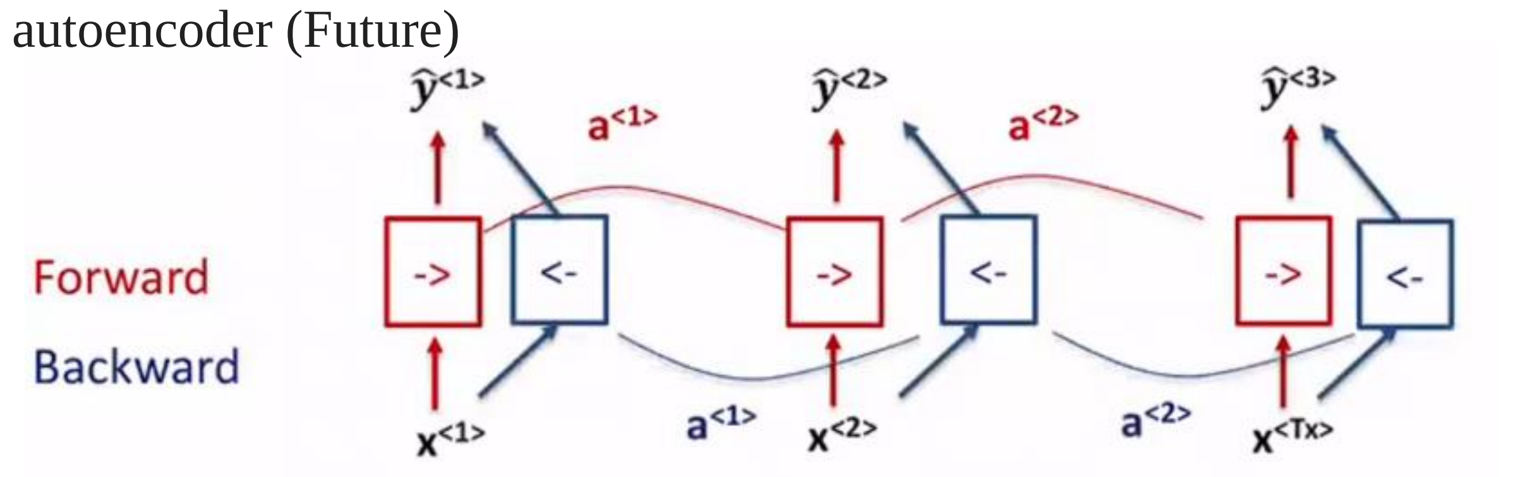


Fig. 10. Bidirectional Layer Structure[5]

Keras.Bidirectional layer is not supported in HLS4ML. We are currently at the beginning stage of implementing this layer.

- Current Status: Implement the converter for the bidirectional layer in HLS4ML

Research Teams

- University of Washington Hardware Development Team
  - Deploy the deep learning model on an ultra low-latency, low-power FPGA and connect the FPGA with the headstage
- University of Washington Neural Interface Team
  - Acquire data from the subject and optimize the deep learning model
- University of Washington Neural processing Algorithms Team
  - Implement algorithms to simplify the data preparation

Acknowledgements & References

- Thanks to the UW Hardware team and the Neural teams
- Thanks to the HLS4ML community
- Thanks to Dr. Elham E Khoda, Dr. Javier Duarte, Dr. Nhan Tran, and Dr. Vladimir Loncar,

- [1] Orsborn A, Shlizerman E, Dadarlat M. (2021) "Understanding & Interfacing with the brain: challenges and opportunities"
- [2] Cartoon Monkey, FPGA Picture, accessed November 2021, <www.shutterstock.com>
- [3] Headstage, <Amy Orsborn's Lab>
- [4] Pandarinath, Chethan, et al. "Inferring single-trial neural population dynamics using sequential auto-encoders." Nature methods 15.10 (2018): 805-815.
- [5] Shlizerman, Eli. "Practical Introduction to Neural Networks." Introduction to Deep Learning Applications and Theory, University of Washington. Lecture.