Empowering AI Implementation: The Versatile SLAC Neural Network Library (SNL) for FPGA

Abhilasha Dave – adave@slac.stanford.edu
Julia Gonski, Kenny Jia, Ryan Herbst, J.J. Russell, Angelo Dragone

CPAD 2024 - Nov. 21, 2024
RDC 05 Parallel Session

U.S. DEPARTMENT OF ENERGY | Stanford University

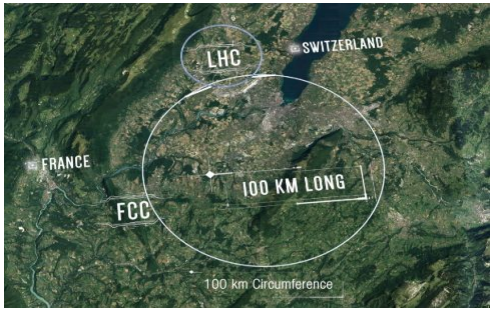
SLAC NATIONAL ACCELERATOR LABORATORY

Growing Challenges in High-Rate Data Processing

Future Needs in Particle Physics and Photon Science

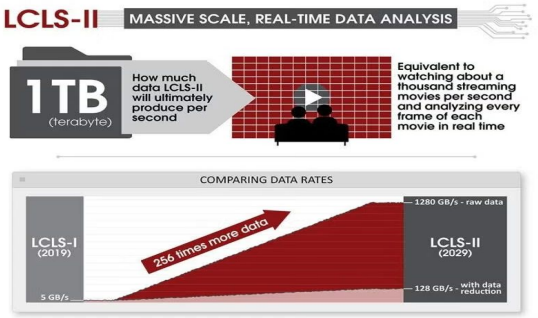
Current LHC Data Rates

- High-energy physics experiments produce vast data at ultra-fast rates. LHC data sampling occurs at 40MHz reaching approximately **1TB/s**



Future Circular Collider at CERN

- The proposed future collider (FCC) aims for **exascale** data rates with multi-stage colliders



LCLS Upgrades at SLAC

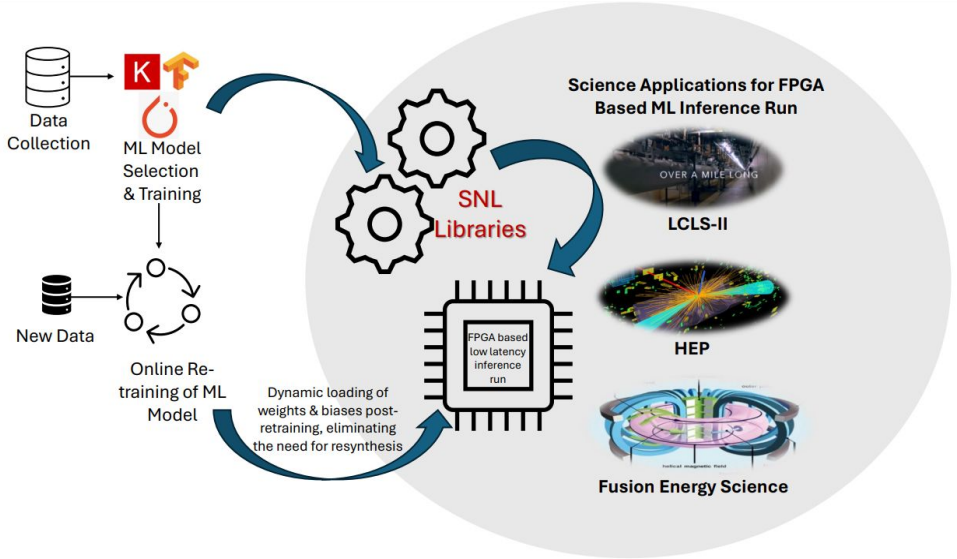
- LCLS will process data at 100 Gb/s from 1MHz pulses

- This vast data flow demands powerful, real-time acquisition and processing systems.
- Future FCC is projected to process data at an exascale rate, challenging detector occupancy and precision with each increase in interaction.
- Photon science also faces similar challenges. SLAC's LCLS X-ray laser, for instance, will need to process high-fidelity data at around 100 Gb/s, creating a parallel demand for innovation in real-time data acquisition and modeling.

SLAC Neural Network Library (SNL)

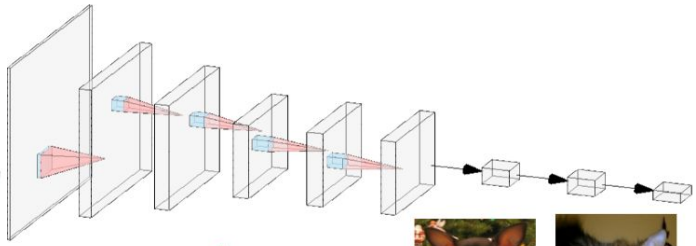
Goals of SNL

- Provide a set of libraries to synthesize AI inference networks into FPGAs
- SNL supports network of medium size:
  - **~100s of thousand** trainable **parameters**
  - Total end to end latency of **~1us to 10ms**
- Pipelined implementations targeting a frame rate of **100KHz – 1MHz**
- **Dynamic reloading of parameters**(weights & biases) that avoids resynthesis
- Supports **Keras like API** for layer definition and configuratio
- Allows standard interface such as **HDF5** for storing weights and biases
- Allow **modular** approach with ability to impliment new and **custom layers**



SNL: https://arxiv.org/pdf/2305.19455

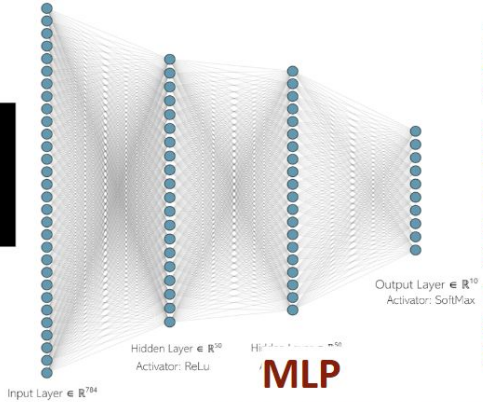
Buildable Neural Networks



CNN/DNN

Dog

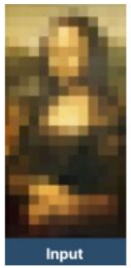
Cat

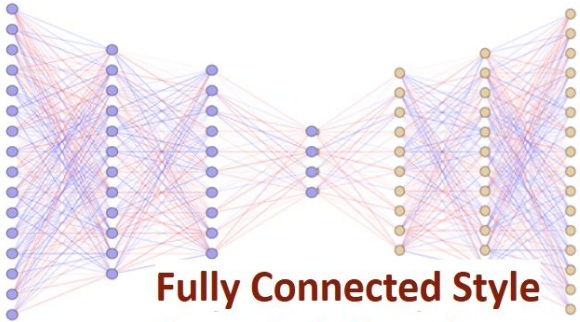


MLP

10 classes

0
1
2
3
4
5
6
7
8
9





Fully Connected Style
Deep AutoEncoders

SNL continuously evolving library

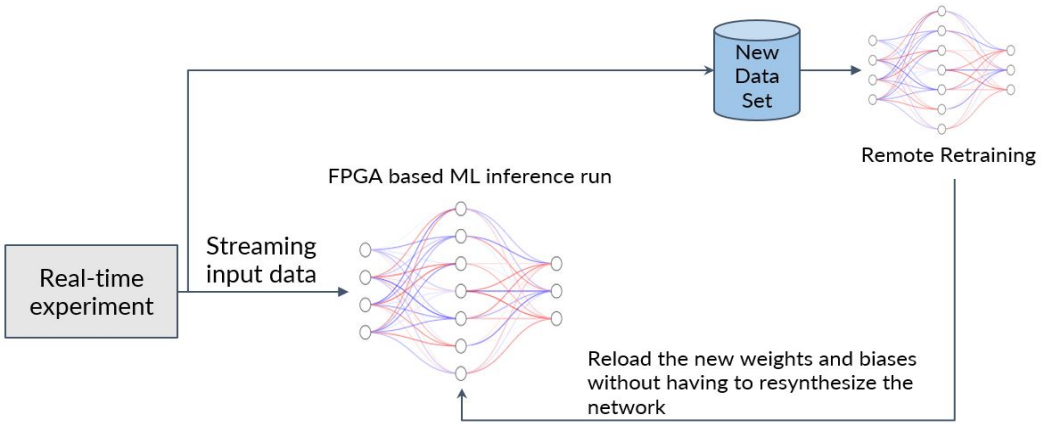
Supported Libraries:

- **NN Layers:** Conv2D, MaxPooling2D, Average Pooling 2D, **Conv1D, MaxPooling1D, Average Pooling 1D,** Dense, Reservoir
- **Activators:** Leaky Relu, **PReLu**, ReLu,SoftMax, **Linear**
- **Data Type:** Fixed Point, Integer, Floating Point

AI generated image

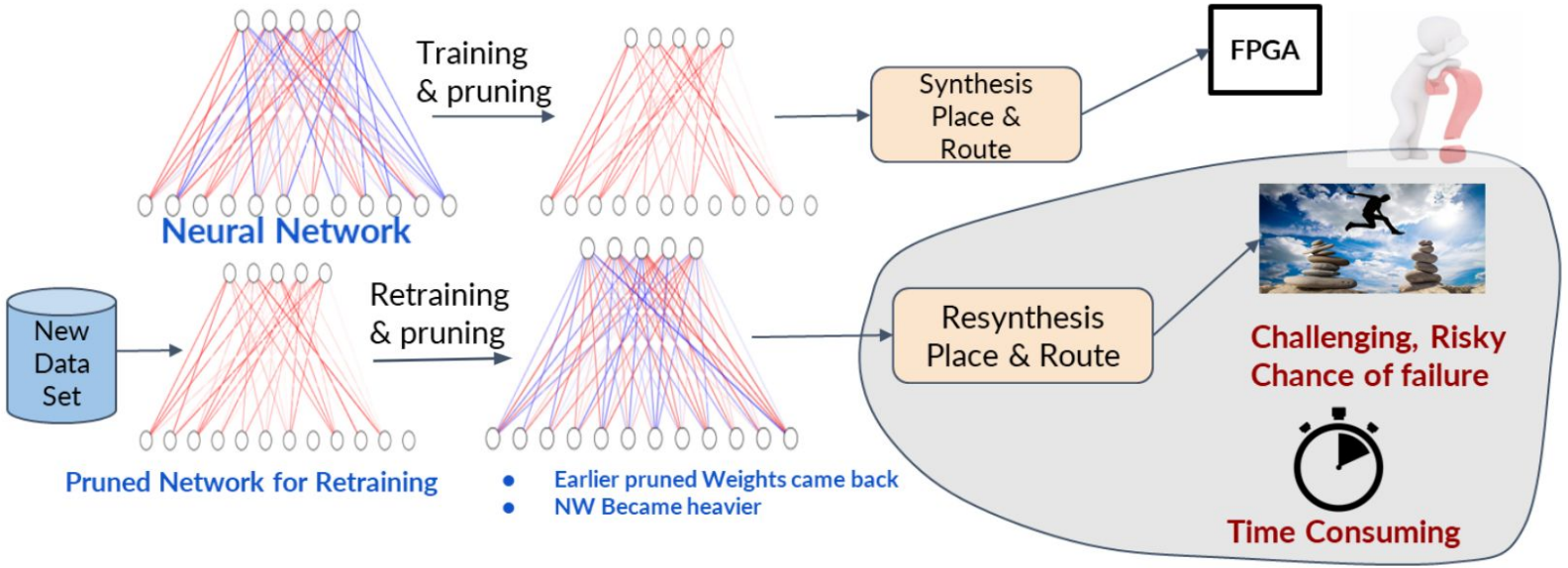
Why Dynamic Loading of Weights and Biases

- Our SNL implementation is targeting **scientific instruments** which will continuously adapt to new data and changing environments
  - **High speed training** to supports this goal
  - Bias and weight **updates in real time**
- Some AI-to-FPGA frameworks take the weights and biases as an input, pruning portions of the network structure to save resources
  - **Re-synthesis is required** for each new training set
  - **Risk of the FPGA implementation failing** due to increase resources usage, timing failures or massive change in internal interconnect structure
- **Large FPGA designs can take hours to days** for the HLS -> synthesis -> place and route cycle to complete
- This is a trade off between **robustness vs. latency** and resource usage



Neural Network Pruning --Incompatible with Dynamic Reloading

- Some AI-to-FPGA frameworks take the weights and biases and pruning portions of the network structure to save resources
  - **Re-synthesis** is required for each new training set
  - **Risk of the FPGA implementation failing** due to increase resources usage, timing failures or massive change in internal interconnect structure



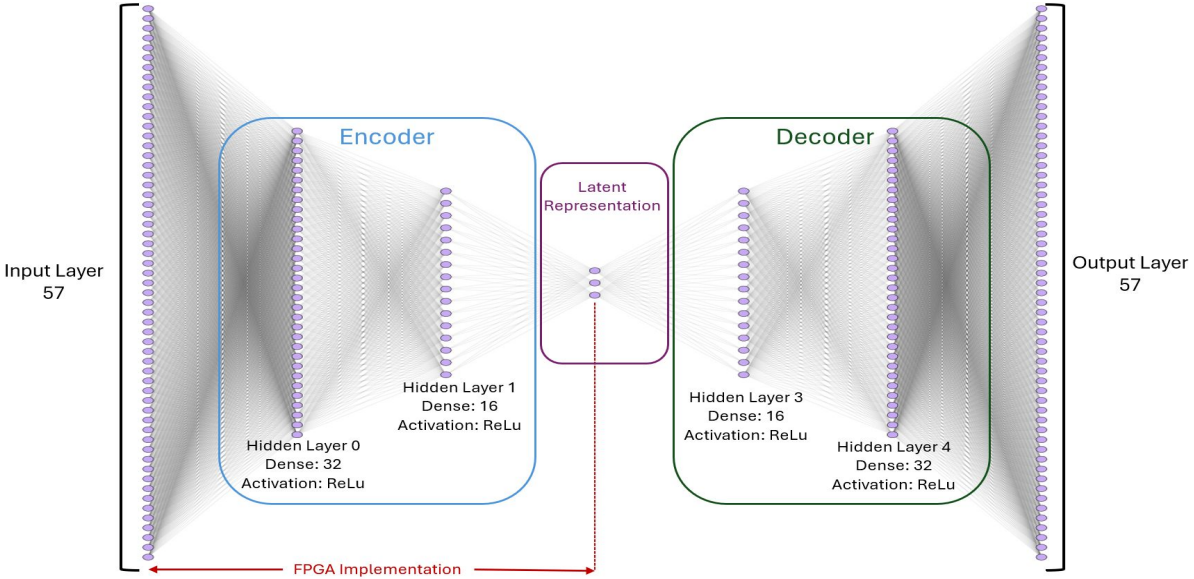
Real-time Anomaly Detection Algorithm for a Collider Trigger System

Role of Variational Autoencoder

- VAEs learns patterns in collider data, allowing efficient detection of outliers representing novel events

Benchmark Models for Performance Evaluation

- Used 3 different encoder model shown in table for resource usage for similar latency between HLS4ML and SNL

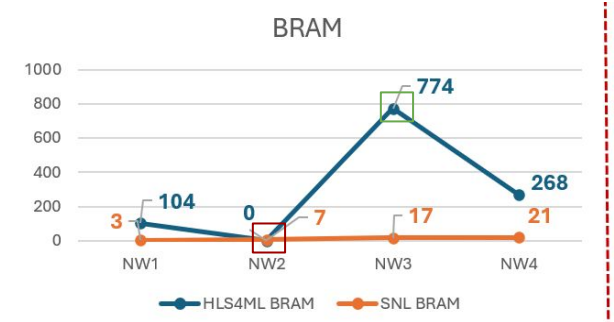


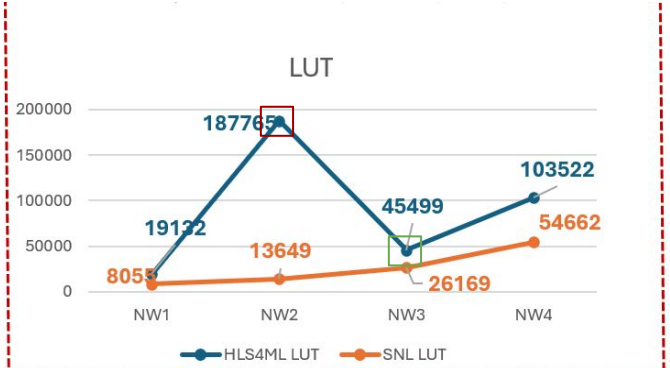
Models	First Layer	Second Layer	Third Layer	Trainable Parameters
Model #1	32	16	3	2435
Model #2	64	32	6	5990
Model #3	128	64	12	16460

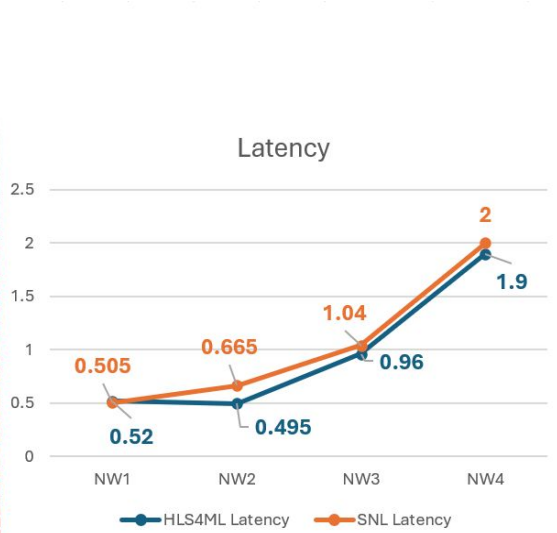
https://arxiv.org/abs/2411.11678
*FPD ATLAS: Julia Gonski, Kenny Jia

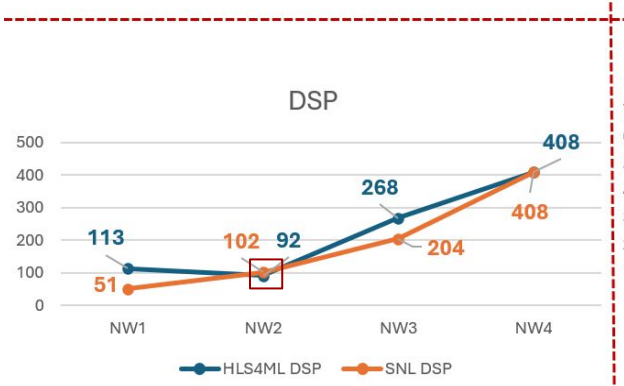
An Evaluation of SNL and HLS4ML Implementation Using Similar Dense Neural Networks AP_FIX<16,8>

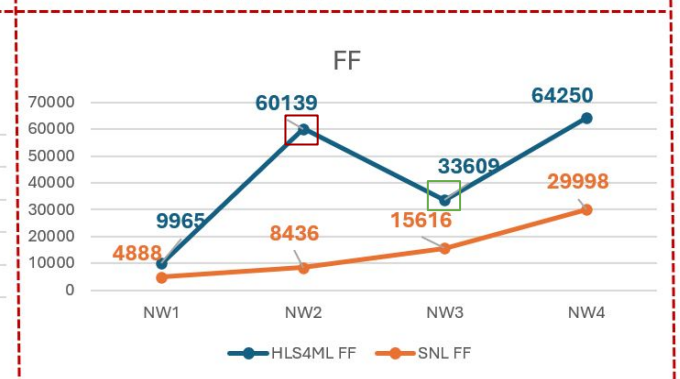
Preliminary Results









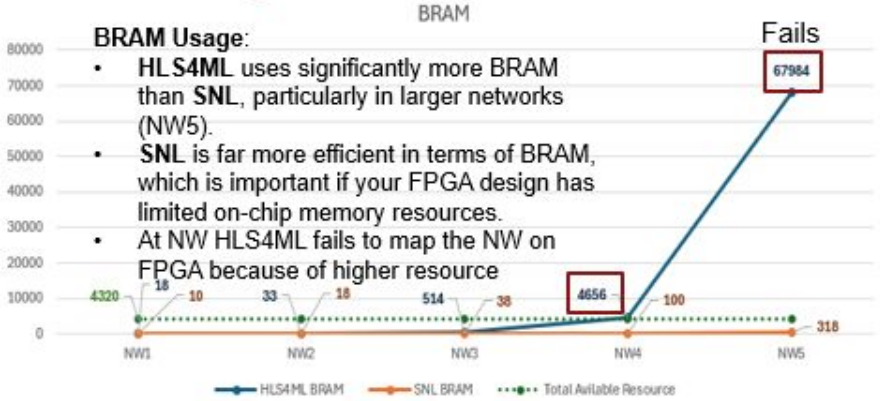


https://arxiv.org/abs/2411.11678

*FPD ATLAS: Julia Gonski, Kenny Jia

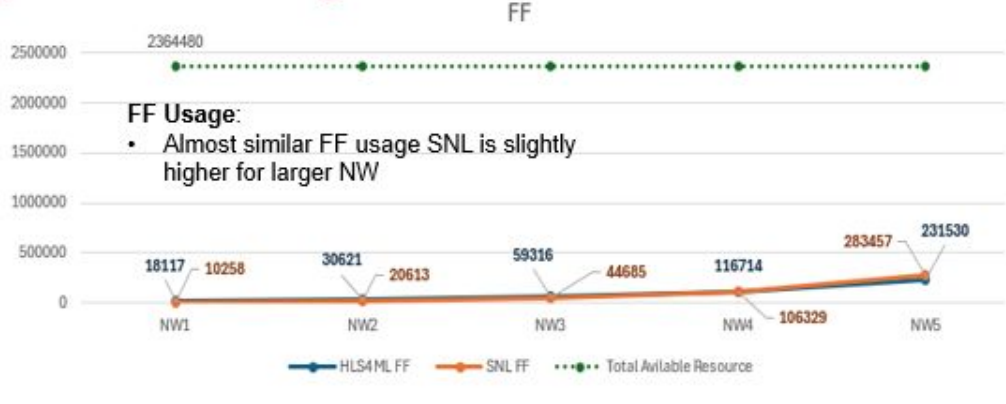
Exploring SNL based ML Inference Through Practical Academic Examples

Preliminary Evaluation of SNL and HLS4ML Implementation Using Similar Dense Neural Networks



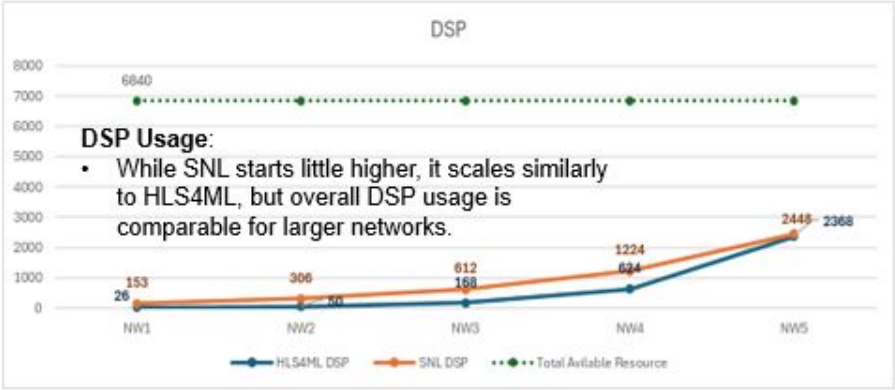
BRAM Usage:

- **HLS4ML** uses significantly more BRAM than **SNL**, particularly in larger networks (NW5).
- **SNL** is far more efficient in terms of BRAM, which is important if your FPGA design has limited on-chip memory resources.
- At NW HLS4ML fails to map the NW on FPGA because of higher resource



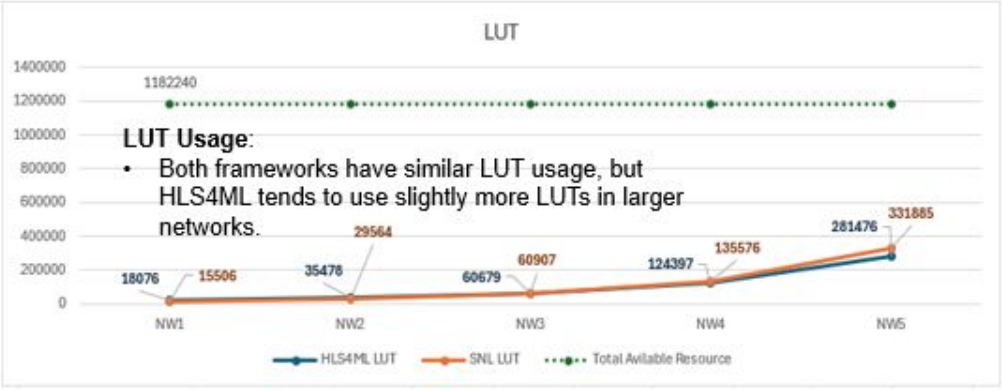
FF Usage:

- Almost similar FF usage SNL is slightly higher for larger NW



DSP Usage:

- While SNL starts little higher, it scales similarly to HLS4ML, but overall DSP usage is comparable for larger networks.

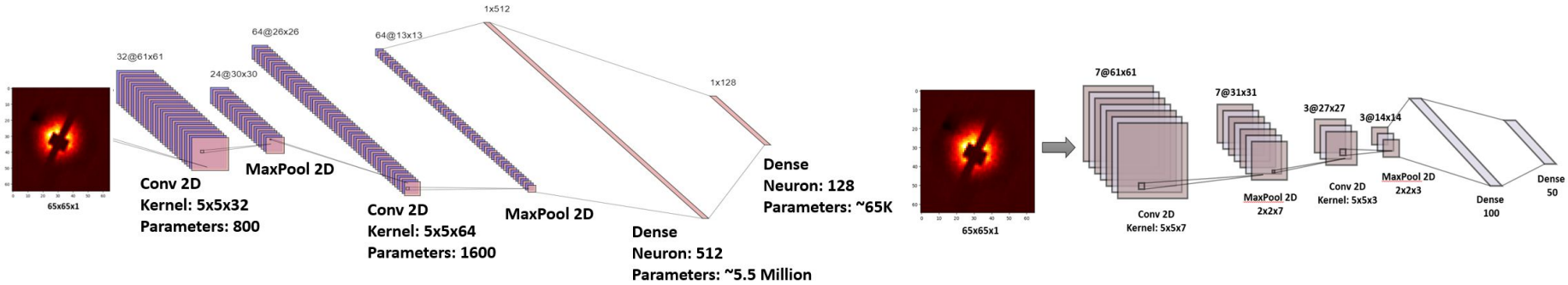


LUT Usage:

- Both frameworks have similar LUT usage, but HLS4ML tends to use slightly more LUTs in larger networks.

SNL Demonstration Model: SpeckleNN

CNN Model Example and Challenges



- Total Parameters:~**5.6Million**
- Data Volume Compression: **98%**
- Classification Accuracy: **98%**

~99% Model size reduction

- Total Parameters:~**56K**
- Data Volume Compression: **98.8%**
- Classification Accuracy: **~94%**

Reference(s):Dave, Abhilasha. et al., 2024, submitted to Frontiers

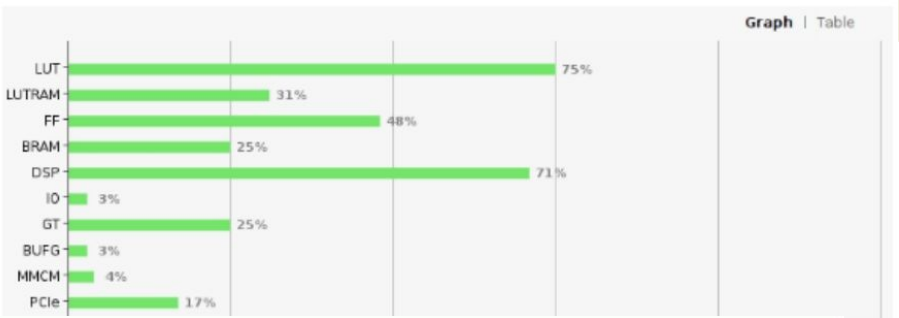
*LCLS Data Analytics: Cong Wang

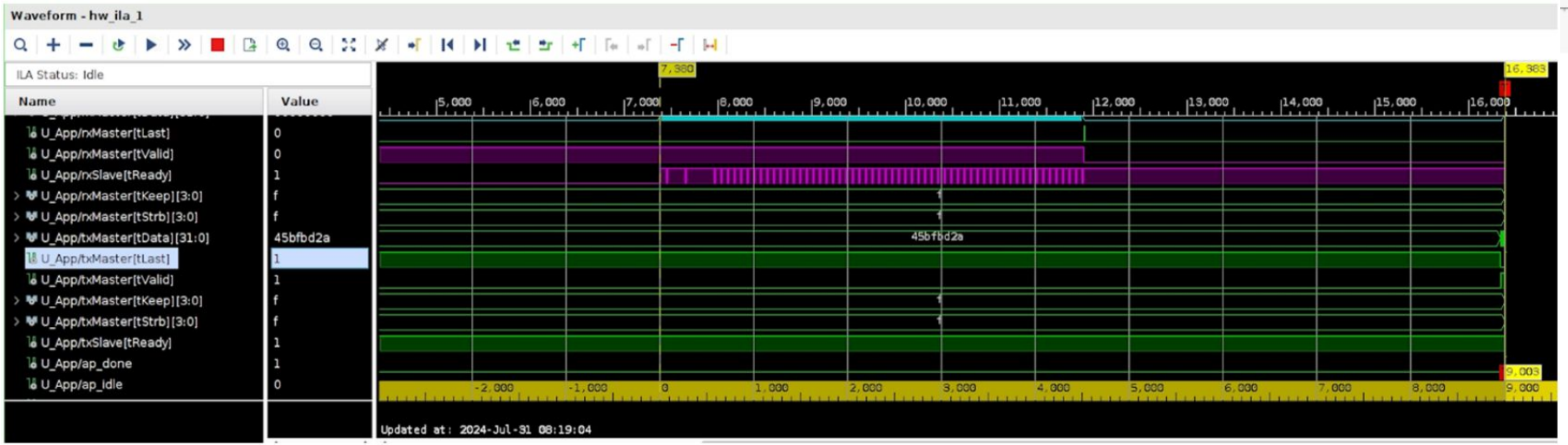
FPGA Resource Utilization

KCU1500 Board

Resource Utilization Analysis: SNL and HLS4ML for FPGA Inference Runs

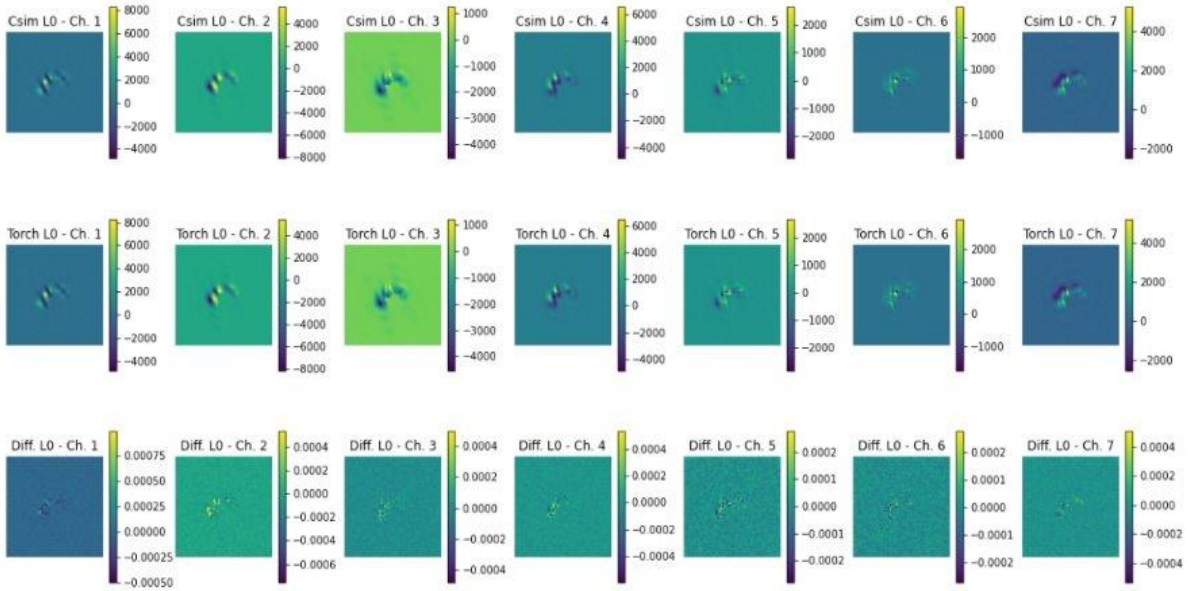
Inference Run Framework	Latency/image (us)	Power (W)
SNL-based FPGA SpackleNN	45.05 (5ns clock period)	9.4
GPU A100	400	73





Reference(s):Dave, Abhilasha. et al., 2024, submitted to Frontiers

Comparison of each Layer Type



Key Observation

- **High Consistency:** Both frameworks produced nearly identical feature maps across all channels.
- Minor differences were observed, most notably in Channel 3, where the difference between the two frameworks was slightly larger but still within a minimal range (around -0.0004 to +0.0004)
- **Minor Numerical Differences:** The pixel wise differences were very small, mostly due to floating-point
- precision differences, with most of the differences falling within the range of -0.0005 to 0.0005
- indicating that the two frameworks are well-aligned in their convolution computation

Reference(s):Dave, Abhilasha. et al., 2024, submitted to Frontiers

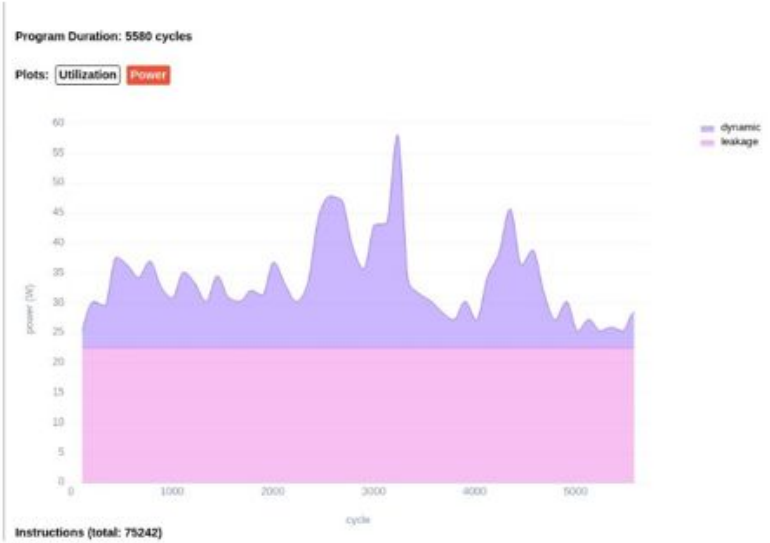
Preliminary Groq Results for SpeckleNN Example

```
(groqflow) adave@rdsrv420:~/speckleNN_Example$ python speckleNN.py
/home/adave/.local/lib/python3.10/site-packages/sklearn/utils/__in
3.0 is required for this version of SciPy (detected version 1.21.6
  from scipy.sparse import issparse



Building "speckleNN"
    ✓ Exporting PyTorch to ONNX
    ✓ Optimizing ONNX file
    ✓ Checking for Op support
    ✓ Converting to FP16
    ✓ Compiling model
    ✓ Assembling model

Woohoo! Saved to ~/.cache/groqflow/speckleNN
Calculated Time: 0.0007321834564208984
Your build's estimated performance is:
0.0000266 seconds
37655.9 inferences per second
Performance estimation completed.
(groqflow) adave@rdsrv420:~/speckleNN_Example$
```



Resource Utilization Analysis: SNL and HLS4ML for FPGA Inference Runs

Inference Run Framework	Latency/image (us)	Power (W)
SNL-based FPGA SpackleNN	45.05 (5ns clock period)	9.4
GPU A100	400	73
Preliminary Groq System Time	732.18	~57

*LCLS SRD AMO: Benjamin Mencer

Summary & Challenges

Summary:

- We compared latency and power consumption of SpeckleNN on an NVIDIA A100 GPU.
- Our SNL-based FPGA-accelerated SpeckleNN achieved:
  -- 8.9x faster latency
  -- 7.8x lower power consumption than the GPU.
- HLS4ML is effective for small neural networks, achieving lower latency, but struggles with larger networks.
- SNL demonstrates promising results in implementing larger neural networks on FPGAs, enabling more complex models to run on hardware.
  - By dynamically loading weights and biases, SNL eliminates the need for complete re-synthesis and simplifying deployment.

Challenges:

- What is the most effective way to manage arithmetic precision in quantized versions of these ML models to ensure consistency across all software and hardware frameworks?
- Additionally, it's crucial to recognize that overall system latency is influenced by the rate at which data is fed into the neural network, as processing cannot exceed the input arrival rate.
- Preliminary tests on Groq, GPU, and FPGA hardware show challenges in consistent performance measurement across different systems