StreamBrain: An HPC DSL for Brain-like Neural Networks on Heterogeneous Systems

Artur Podobas*, Martin Svedin*, Steven W. D. Chien*, Ivy B. Peng+,
Naresh B. Ravichandran*, Pawel Herman*, Anders Lansner*,**, Stefano Markidis*
Email: podobas@kth.se

*KTH Royal Institute of Technology, Sweden
+Lawrence Livermore National Laboratory, USA
** Stockholm University, Sweden

Introduction

- Streambrain, a Keras-like library for implementation, evaluation, and deployment of BCPNN for use in future high-performance computers and data-centers,
- Analysis, implementation, validation, and empirical evaluation of three different BCPNN backends for CPUs, GPUs, and FPGAs, on the MNIST data-set,
- Empirical evaluation on both batching (a well-known deep-learning optimization) and variable-precision arithmetics on the BCPNN model.

Bayesian Confidence Propagation Neural Network

Brain-like Bayesian Confidence Propagation Neural Network (BCPNN) is a probabilistic graphical model that employs graphs to represent a problem by combining probability and graph theories. It models the problem with a collection of random variables as joint distribution. Each node of the graph represents a random variable while the edge represents the dependence or correlation between the variables. A training phase determines the weights (w) and biases (b) characterizing the connection. Refer to Ravichandran et al.[1] for the governing equations.

StreamBrain DSL

A BCPNN model can be conceived as a stack of layers that individually takes in an input array and gives an output array just as regular neural networks do. Inspired by existing neural network frameworks, we opted to create a Keras-like interface for StreamBrain, to capture the simplicity.

```
# 1. Create empty network
model = BCPNN.Network(...)
# 2. Add layers
model.add(BCPNN.StructuralPlasticityLayer(..))
model.add(BCPNN.DenseLayer(...))
# 3. train and evaluate
model.fit(dataset=(...))
model.evaluate(dataset=(...))
```

Figure 1:A simple BCPNN Network through the StreamBrain framework

Heterogeneous Backends

- **FPGA Backend**
  - Described in Intel OpenCL SDK for FPGAs
  - Supports custom, non-IEEE-754 conforming, floating-point representations using FloPoCo RTL components directly from OpenCL
  - Reaches between 150 and 223 MHz of clock frequency

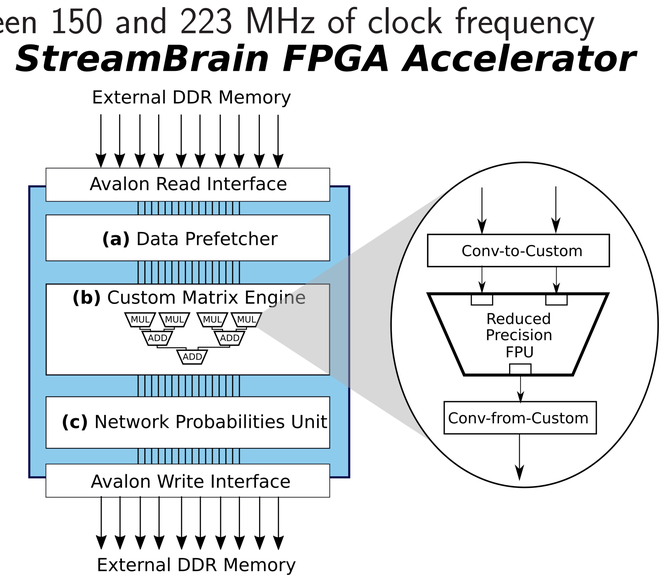

Figure 2:Conceptual illustration of the StreamBrain BCPNN FPGA accelerator, showing key operational components: (a) The prefetcher that brings the neural network into the accelerator and (b-c) the custom computational units that operate using custom FloPoCo HDL FPUs modules.

- **GPU Backend(s)**
  - SB-GPU (Partial): Offloads compute-intensive functions to the GPU
  - SB-GPU (Full): Implements the entire StreamBrain on the GPU
- **CPU Backend(s)**
  - Vectorized OpenMP version that leverage Intel-MKL

Evaluation

Evaluation is performed on a FPGA-system with Intel DE5-Net board, and a GPU-system with NVIDIA V100 GPUs. The training and testing uses the MNIST dataset for character recognition.

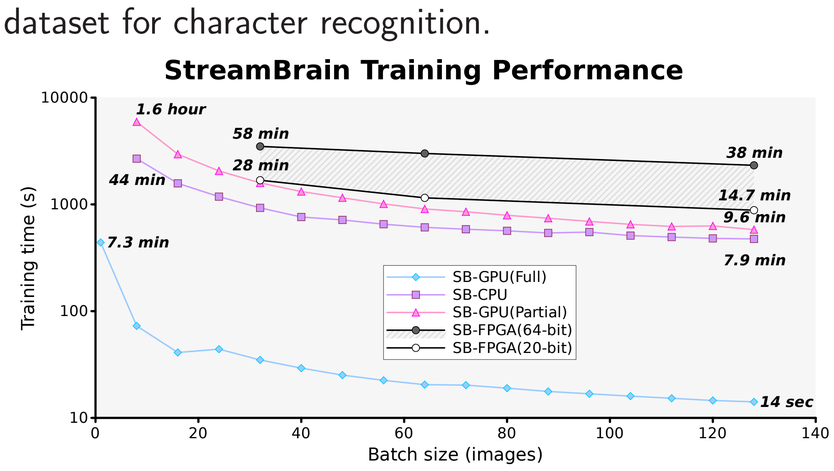

Figure 3:StreamBrain training performance of MNIST using the different implementations when running on the Kebnekaise machine and on the FPGA, showing the total execution time (y-axis) as a function of the batch-size (x-axis). The shaded area shows the performance of the SB-FPGA when varying the number representation in bits.

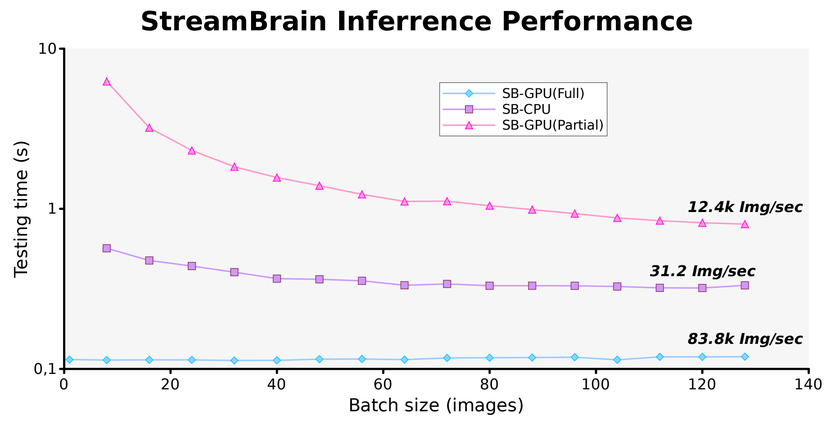

Figure 4:Figure showing the inference performance of the StreamBrain implementations when running on the Kebnekaise machine, showing the execution time (y-axis) as a function of batch-size (x-axis).

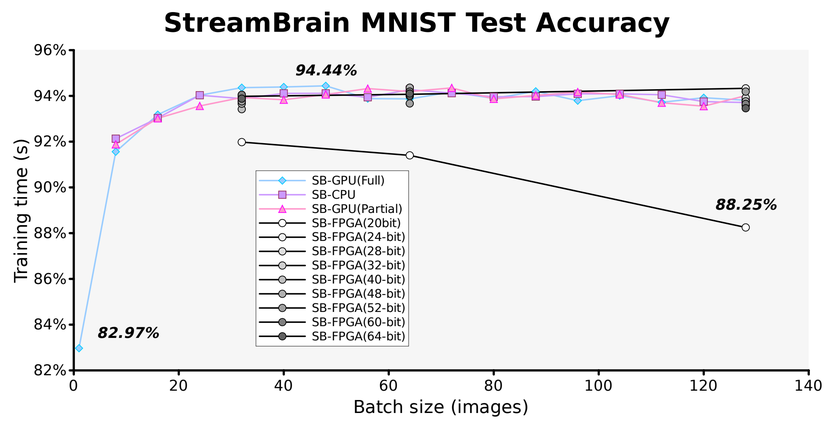

Figure 5:Figure showing the testing accuracy (y-axis) of the different Streambrain versions and different batch-size (x-axis).

Conclusions

- We designed and developed **Streambrain**: a DSL for BCPNN models on heterogeneous systems including FPGAs, GPUs, and CPUs.
- We evaluated the performance and accuracy, demonstrating that training the well-known MNIST data-set can be as fast as 15 seconds.
- We showed that batching is also adaptable inside BCPNN, allowing computational intensity to be controlled.

References

[1] Ravichandran, Naresh Balaji, Anders Lansner, and Pawel Herman. "Learning representations in Bayesian Confidence Propagation neural networks." arXiv preprint arXiv:2003.12415 (2020).

Acknowledgement

Funding for the work is received from the European Commission H2020 program, Grant Agreement No. 801039 (EPiGRAM-HS). Experiments were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at HPC2N.