

# Approximate Computation using Neutralized FPU

Schuyler Eldridge\*, Florian Raudies†, Ajay Joshi\*

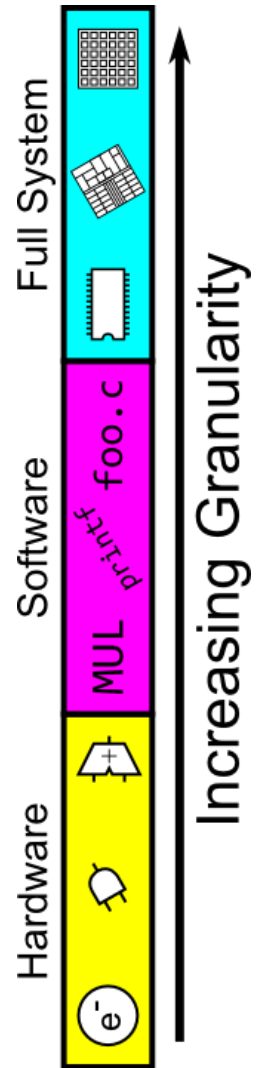
\*Electrical and Computer Engineering, Boston University

†Center for Computational Neuroscience and Neural  
Technology, Boston University



# Granularities of Computation

- Within the realm of computing, there exists many computational granularities encompassing:
  - Hardware
  - Software
  - Full Systems
- Power, performance, and area benefits can be achieved by approximating granularities of computation
- One approach towards approximating computation uses neural networks as function approximators



# NNs as Approximators – Gen. Purpose

---

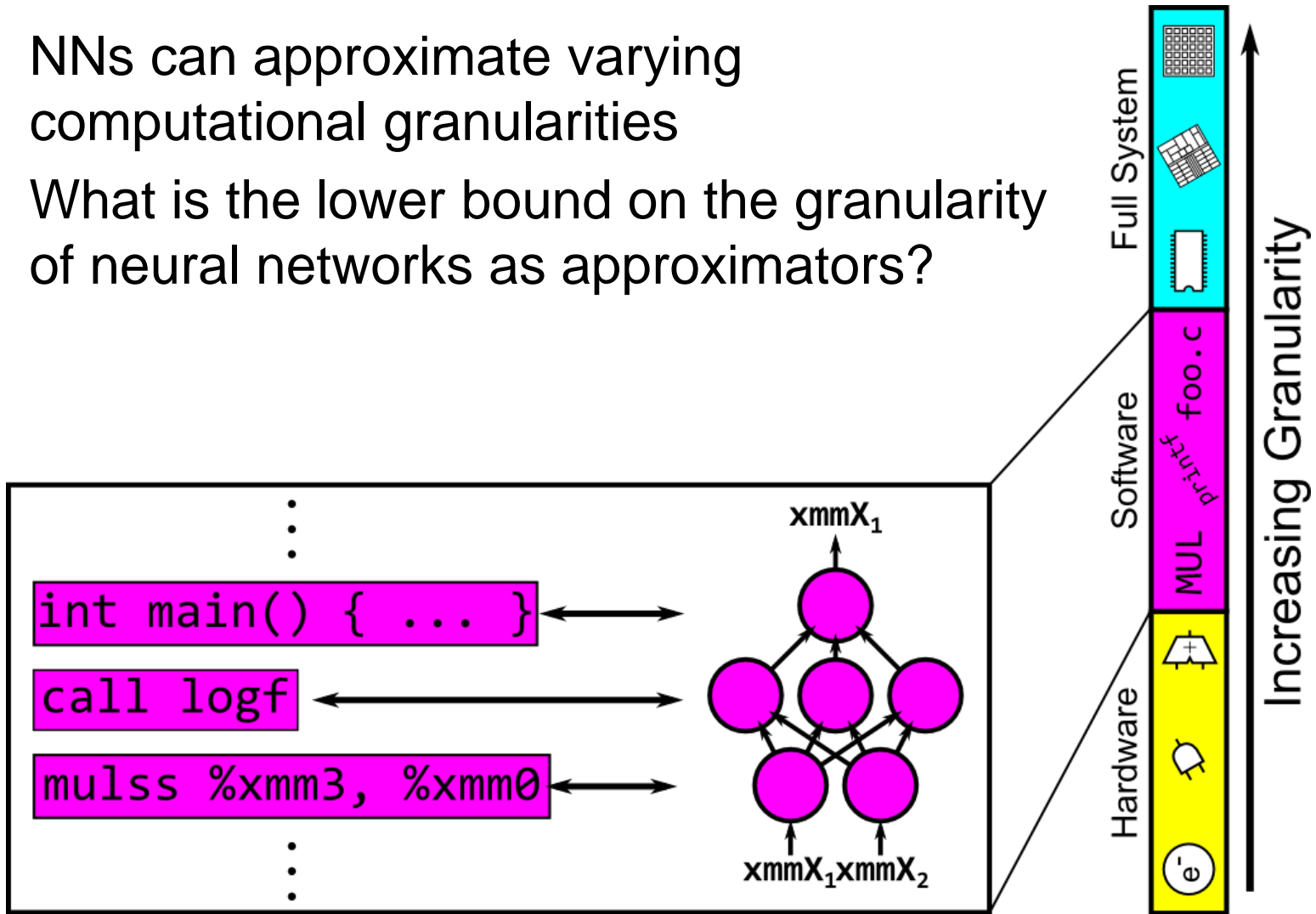
- Neural Networks can be general purpose function approximators
  - NNs can approximate regions of code [1]
  - NNs can approximate dominant functions in PARSEC [2]
  - Hardware implementations of NNs impart performance benefits
  - Accuracy impacts are acceptable
- Approximating large regions of code is best
  - Maximal performance benefits are imparted by large code regions
  - This can be achieved via compiler assisted user annotations [1]
  - This can be achieved via explicit user modifications [2]
  - User involvement is expensive and difficult

[1] H. Esmailzadeh et al., “Neural acceleration for general-purpose approximate programs,” in IEEE MICRO, 2012.

[2] T. Chen et al., “Benchnn: On the broad potential application scope of hardware neural network accelerators,” in IISWC. IEEE, 2012, pp. 36–45.

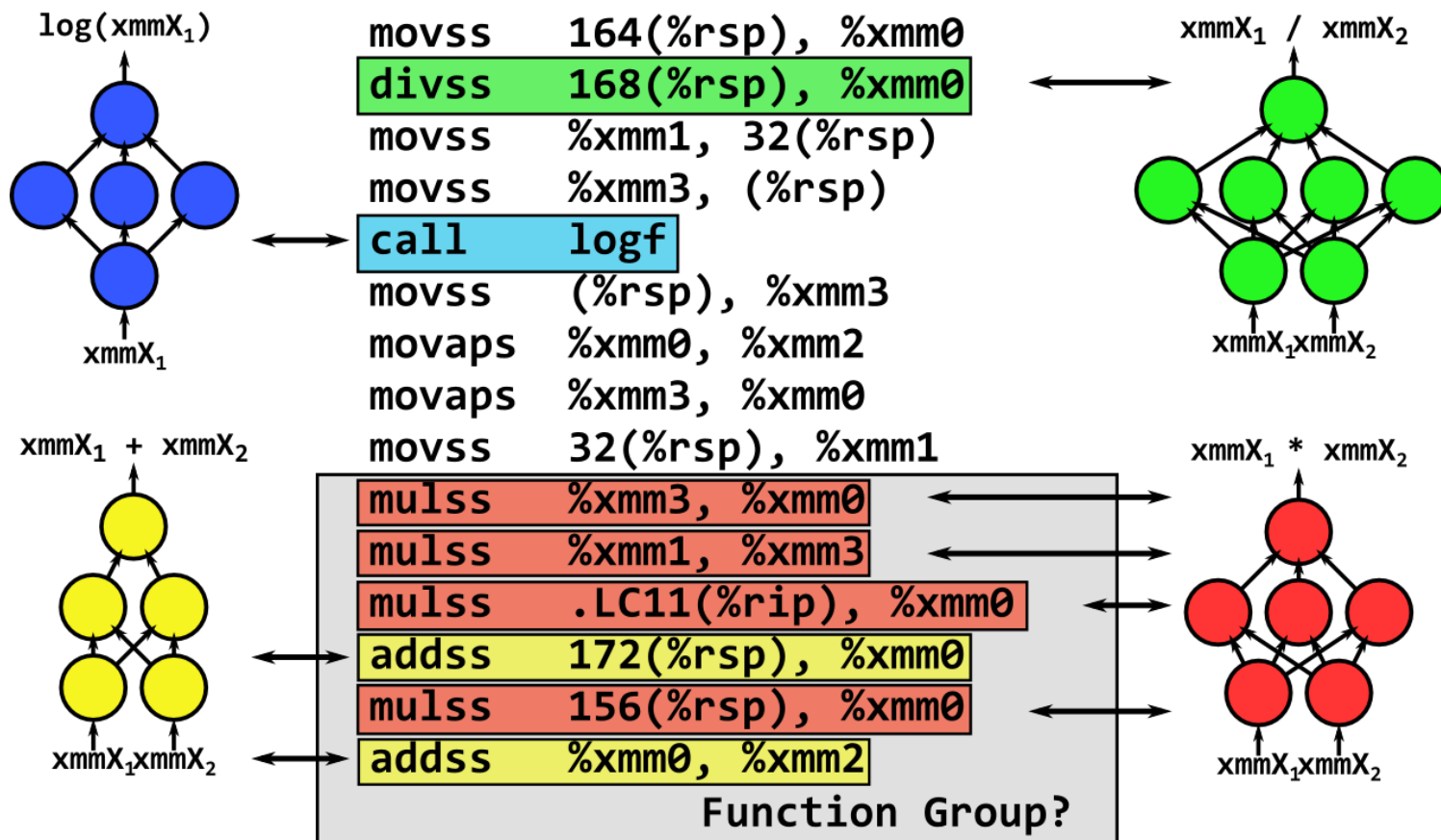
# NNs as Approximators - Granularity

- NNs can approximate varying computational granularities
- What is the lower bound on the granularity of neural networks as approximators?



# NNs as Approximators – Example

- NNs can approximate different instructions
- Approximation occurs in general purpose code



# Neutralized FPMul – Details

---

- Can all instructions be approximated?
  - Instructions can be conditionally or generally approximated
- We explore approximating floating point multiplications
- As a case study, all floating point multiplications in three PARSEC [3] benchmarks (blackscholes, bodytrack, swaptions) are replaced with neutralized versions

[3] Bienia, C. et al., “The parsec benchmark suite: Characterization and architectural implications,” in 17<sup>th</sup> International Conference on Parallel Architectures and Compliance Techniques. ACM, 2008, pp. 72-81.

# Neutralized FPMul – Implementation

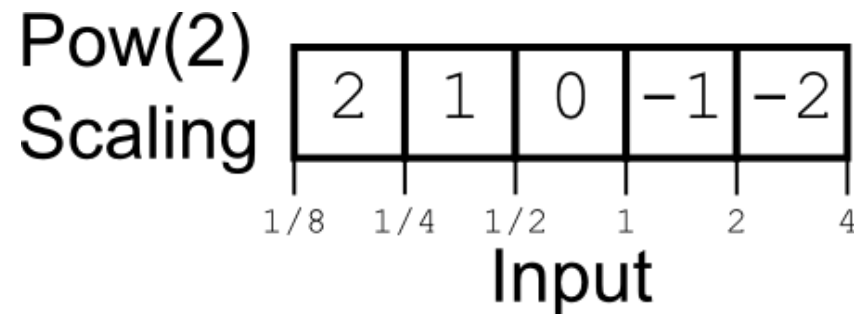
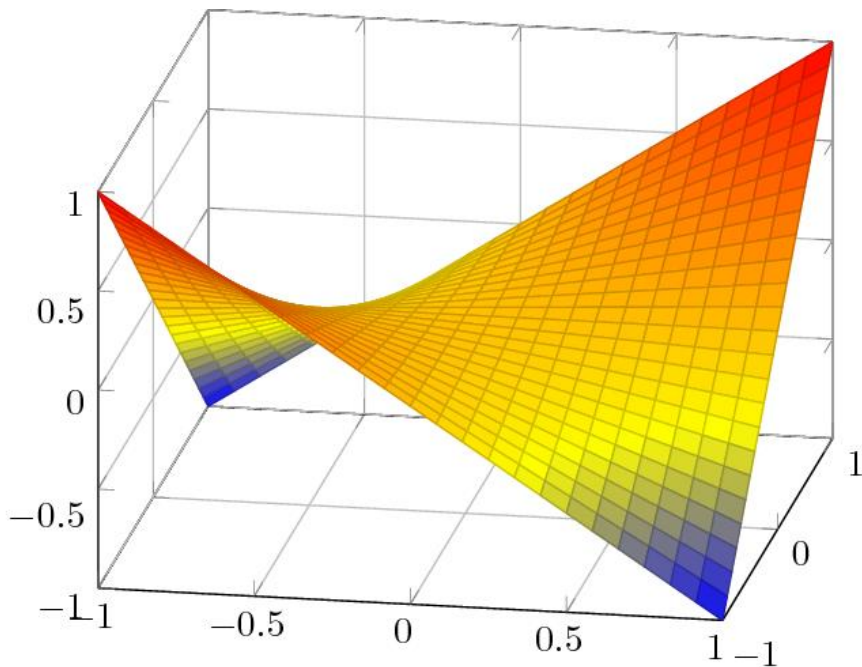
---

- Floating point multiplications are approximated with MLP networks
- MLPs are implemented using the Fast Artificial Neural Network Library (FANN) [4]
- The range of floating point numbers is problematic
  - Floating point range is large
  - Floating point numbers require accuracy for both large and small numbers
  - An MLP cannot approximate this range well for all values

[4] “Implementation of a fast artificial neural network library (fann),” Department of Computer Science University of Copenhagen (DIKU), Tech. Rep., 2003, <http://fann.sf.net>.

# Neutralized FPMul - Scaling

- Multiplication are approximated on range:  $[-1, 1]$
- Inputs are scaled by powers of 2 onto the disjoint range:  $[-1, -0.5]$ ,  $[0.5, 1]$
- Different scalings are suitable for other functions





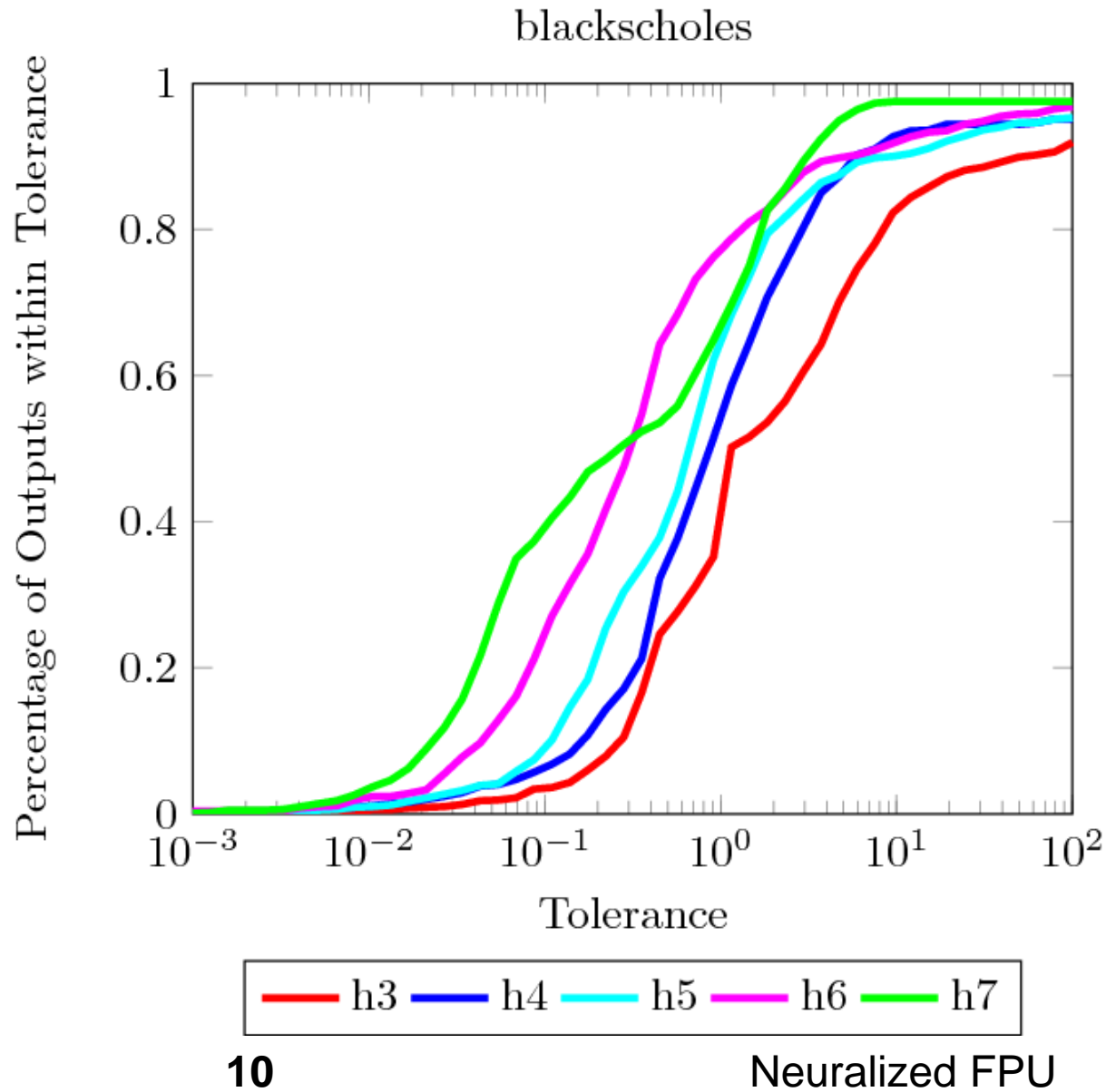
# Neuralized FPMul – Training/Testing

---

- Multiplication MLPs are trained with resilient backpropagation using training and validation datasets
- Mean Squared Error (MSE) minimization was the goal of training
- The design space is explored from between 3 to 7 hidden nodes
- Blackscholes, bodytrack, and swaptions are executed using this Neuralized FPMul
- Output accuracy is measured
- An accurate output is defined as being within a percent tolerance of a correct output

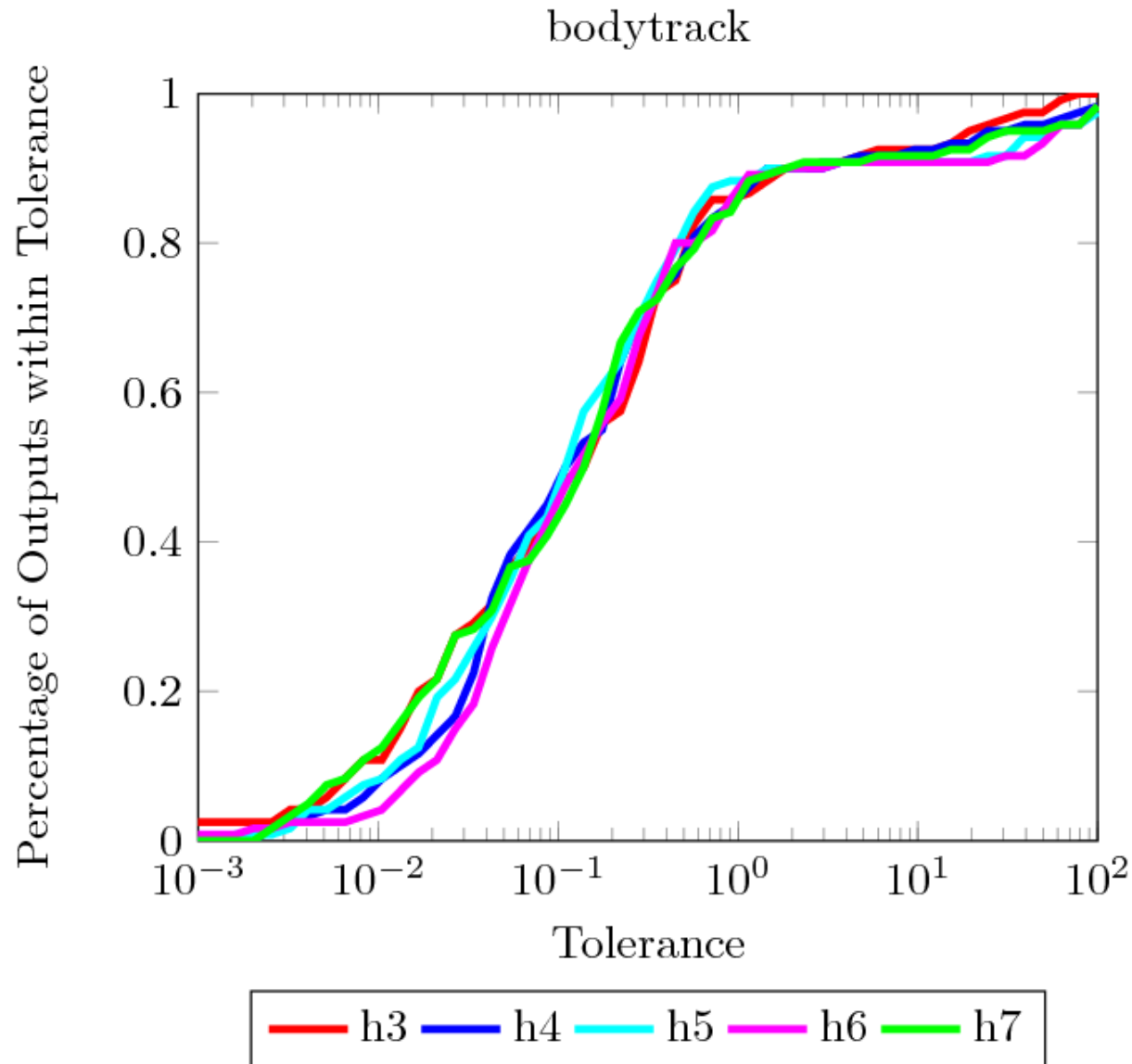
# Results - blackscholes

- Blackscholes is dependent on the accuracy of the network approximating multiplication
- Networks with more nodes and lower MSE outperform networks with less nodes



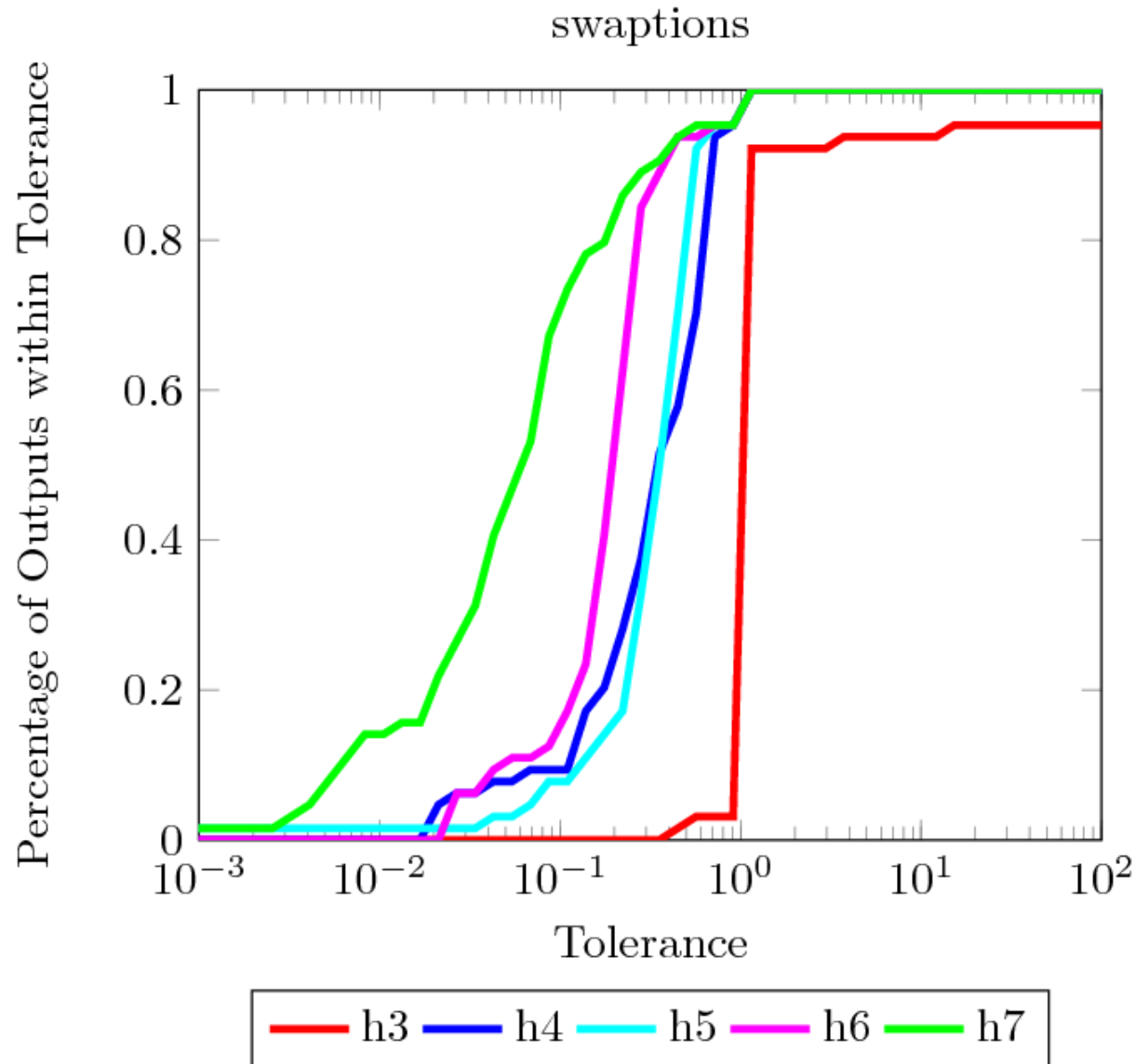
# Results - bodytrack

- The accuracy of bodytrack is independent of approximation
- Bodytrack is a particle filter application
- Percent tolerance may not be the best metric
- More complex networks provide no benefits



# Results - swaptions

- Swaptions shows a dependency on network accuracy
- A 7 hidden node network greatly outperforms a 6 hidden node network
- For high tolerances, network topology is less important



# Discussion - Approximation

---

- One caveat: Not all floating point multiplications can be approximated
  - Bodytrack requires one variable to be on an exact range
  - Floating point multiplications in `glibc` `exp` cannot be approximated
- Coding styles affect the required precision
- Two examples found in PARSEC are:
  - Errors occur when function outputs are out of bounds
  - Errors occur when table lookups are derived from floating point computations
- A contrived example is:
  - Errors occur when control flow is dependent on the result of a floating point multiplication

# Discussion – Impracticalities

---

- Using current hardware neural networks are expensive
  - The total complexity of the approximation network should be less than the complexity of the approximated function
- Implementing neural network approximators using more operations than the instruction being approximated is a poor design choice
- Efficient implementations and emerging technologies may make small granularity practical
- Nevertheless, probing the lower bounds of approximating computational granularities is the goal of this research

# Discussion - Fault Tolerance

---

- An ignored benefit of neural network approximators is fault tolerance
- Fault tolerant training / retraining could be beneficial
- Examples include the following:
  - NNs can train/retrain with CMOS-specific faults [5]
  - Approximators with large granularity are robust in time and space
  - NNs can provide graceful performance degradation
- The best method to leverage fault tolerance remains an open question

[5] O. Temam, “A defect-tolerant accelerator for emerging highperformance applications,” in 39<sup>th</sup> ISCA, June 2012, pp. 356 –367

# Questions

