Neural Network-Based Accelerators for Transcendental Function Approximation

Schuyler Eldridge* Florian Raudies† David Zou*
Ajay Joshi*

*Department of Electrical and Computer Engineering, Boston University
†Center for Computational Neuroscience and Neural Technology, Boston University
schuye@bu.edu

May 22, 2014

This work was supported by a NASA Office of the Chief Technologist's Space Technology Research Fellowship.

Technology Scaling Trends

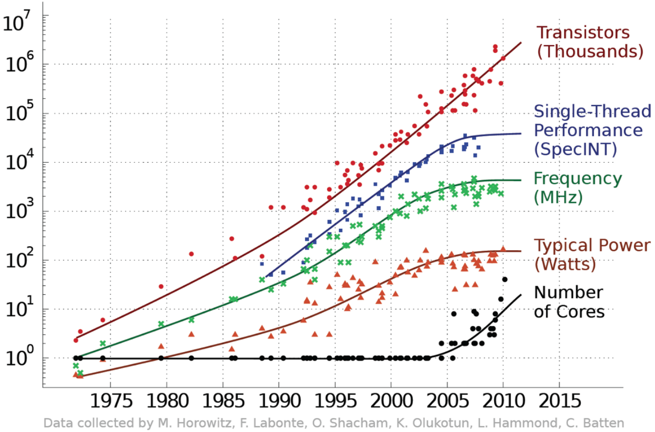

Figure 1: Trends in CMOS technology [Moore et al., 2011 *Salishan*]

Accelerators to the Rescue?

Energy Efficient Accelerators...

- Lessen the utilization crunch of Dark Silicon
- Are cheap due to plentiful transistor counts
- Are typically special-purpose

Approaches to General Purpose Acceleration

- QsCores – Dedicated hardware for frequent code patterns [Venkatesh et al., 2011 *MICRO*]
- NPU – Neural network-based approximation of code regions [Esmaeilzadeh et al., 2012 *MICRO*]

Neural Networks (NNs) as General-Purpose Accelerators

The good and the bad...

- NNs are general-purpose approximators [Cybenko, 1989 *Math. Control Signal*, Hornik, 1991 *Neural Networks*]
- But... NNs are still approximate

Approximation may be acceptable

- Modern recognition, mining, and synthesis (RMS) benchmarks are robust [Chippa et al., 2013 *DAC*]

Library-Level Approximation with NN-Based Accelerators

Big Idea

- Use NNs to approximate library-level functions
  - cos, exp, log, pow, and sin
- Explore the design space of NN topologies
  - Define and use an energy–delay–error product (EDEP) metric
- Evaluate energy–performance improvements
  - Use an energy–delay product (EDP) metric
- Evaluate accuracy of...
  - NN-based accelerators vs. a traditional approach
  - Applications using NN-based accelerators

Multilayer Perceptron (MLP) NN Primer

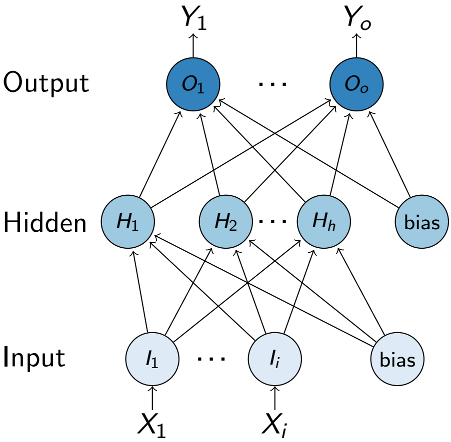

Figure 2: NN with $i \times h \times o$ nodes.

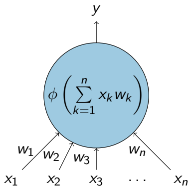

Figure 3: One neuron

Equations

$$y = \phi\left(\sum_{k=1}^{n} x_k w_k\right)$$

$$\phi_{sigmoid} = \frac{1}{1+e^{-2x}}$$

$$\phi_{linear} = x$$

NN-Based Approximation Requires Input–Output Scaling

Approximating Unbounded Functions on Bounded Domains

- NNs cannot handle unbounded inputs
- Input–output scaling can extend the effective domain and range of the approximated function
- This approach is suitable when...
  - A small region is representative of the whole function
  - There exist easy[a] operations to scale inputs and outputs
- Specifically, we use the CORDIC [Volder, 1959 *IRE Tran. Comput.*] scalings identified by Walther [Walther, 1971 *AFIPS*]

[a] By "easy", I mean multiplication with a constant, addition, bitshifts, and rounding.

Walther's Scaling Approach [Walther, 1971 *AFIPS*] for $\exp x$

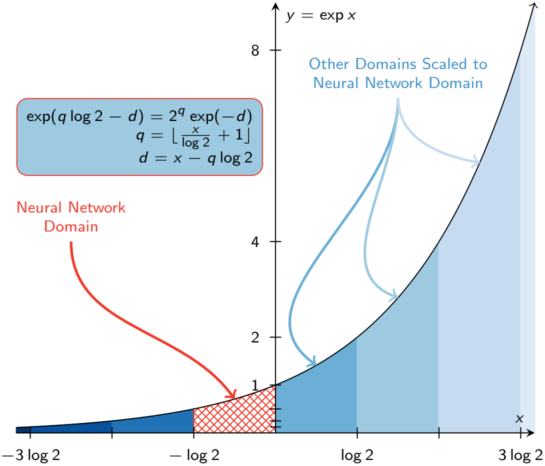

Figure 4: Graphical scaling for $\exp x$

Scaling Steps

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Similar Scalings Exist

- $\cos x$ and $\sin x$
- $\log x$

Walther's Scaling Approach [Walther, 1971 *AFIPS*] for $\exp x$

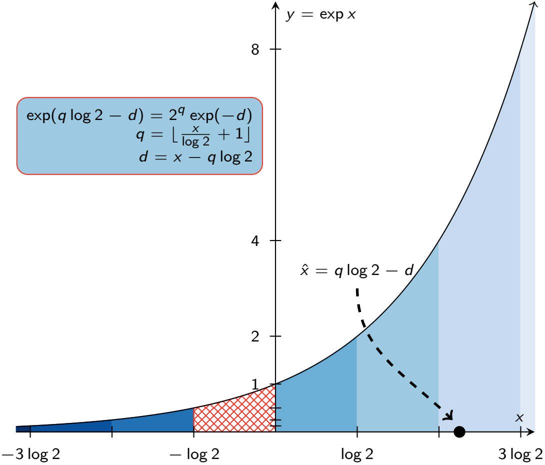

Figure 4: Graphical scaling for $\exp x$

Scaling Steps

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Similar Scalings Exist

- $\cos x$ and $\sin x$
- $\log x$

Walther's Scaling Approach [Walther, 1971 *AFIPS*] for $\exp x$

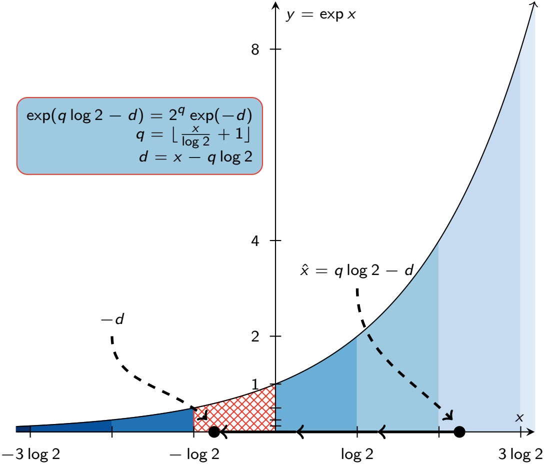

Figure 4: Graphical scaling for $\exp x$

Scaling Steps

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Similar Scalings Exist

- $\cos x$ and $\sin x$
- $\log x$

Walther's Scaling Approach [Walther, 1971 *AFIPS*] for $\exp x$

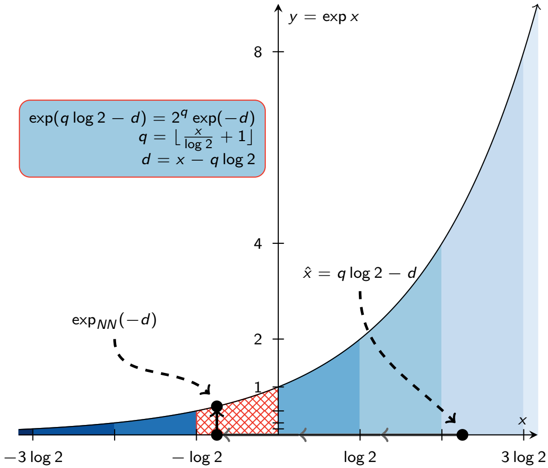

Figure 4: Graphical scaling for $\exp x$

Scaling Steps

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Similar Scalings Exist

- $\cos x$ and $\sin x$
- $\log x$

Walther's Scaling Approach [Walther, 1971 *AFIPS*] for $\exp x$

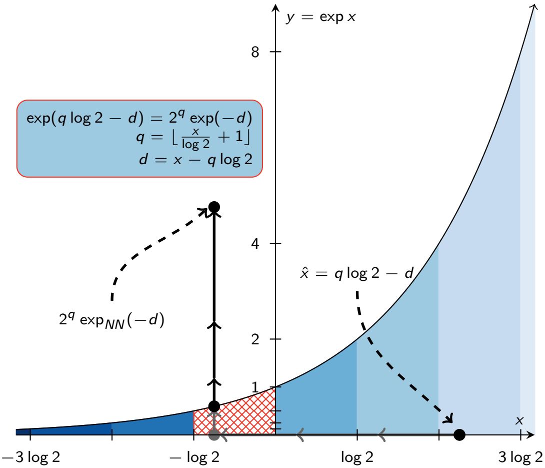

Figure 4: Graphical scaling for $\exp x$

Scaling Steps

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Similar Scalings Exist

- $\cos x$ and $\sin x$
- $\log x$

Walther's Scaling Approach [Walther, 1971 *AFIPS*] for $\exp x$

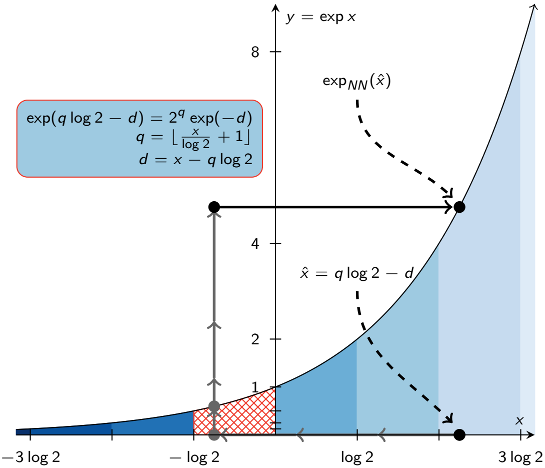

Figure 4: Graphical scaling for $\exp x$

Scaling Steps

1. Scale inputs onto NN domain
2. NN approximates function
3. Scale outputs onto full range

Similar Scalings Exist

- $\cos x$ and $\sin x$
- $\log x$

Fixed Point Accelerator Architecture for $1 \times 3 \times 1$ NN

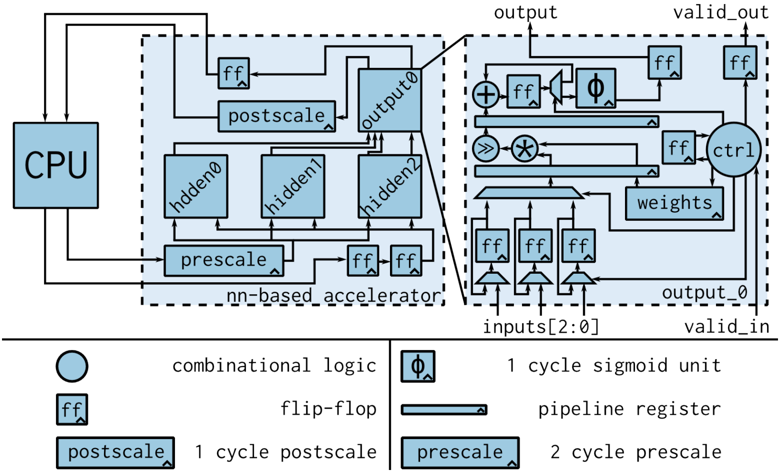

Figure 5: Block diagram of an NN-based accelerator architecture

NN Topology Evaluation – Design Space Exploration

Candidate NN Topologies

- Fixed point
- 1–15 hidden nodes
- 6–10 fractional bits

NN Evaluation Criteria

- Energy
- Performance
- Accuracy

Energy–Delay–Error Product (EDEP)

- Optimal NN topology minimizes EDEP

$$\text{EDEP} = \text{energy} \times \frac{\text{latency in cycles}}{\text{frequency}} \times \text{mean squared error}$$

NN Topology Evaluation – Results

Table 1: MSE and energy consumption of the NN-based accelerator implementation of transcendental functions.

Func.	NN	MSE ($\times 10^{-4}$)	Energy (pJ)	Area (um^2)	Freq. (MHz)
cos	h1_b6	9	8	1300	340
sin	h1_b6	7	8	1300	340
exp	h3_b7	2	25	3600	340
log	h3_b7	1	25	3600	340
pow	h3_b7	432	102	3600	340

Evaluation Notes

- Evaluated with a 45nm predictive technology model (PTM)
- pow computed using exp and log:

$$a^b = \exp\left(b \log a\right)$$

NN Topology Evaluation Results

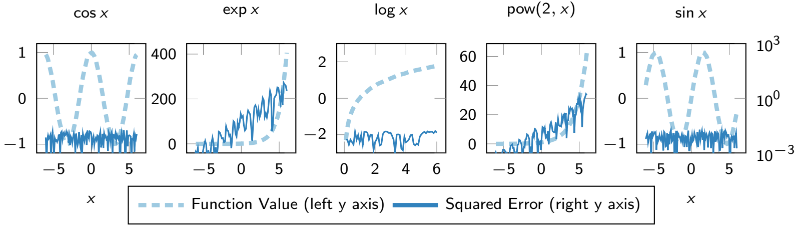

Figure 6: NN-based functions and their errors. Note: Error is plotted on a log scale using the right y axis.

Evaluation Notes

- Functions well approximated by their NNs
- Due to input–output scaling, error is proportional to output value

Evaluation Approach

Approach – Energy

- Determine traditional `glibc` instruction breakdown
- Determine $^{energy}/_{instruction}$ in 45nm PTM
- Determine `glibc` $^{energy}/_{function}$
- Compare traditional and NN-based execution using EDP

Approach – Accuracy

- Replace all transcendental function calls with NNs
- Evaluate application output accuracy

Traditional glibc Instruction Breakdown

Table 2: Mean floating point instruction counts.

Func.	addsd	addss	mulsd	mulss	subsd	subss	Total Instructions
cos	7	0	12	0	8	0	115
cosf	0	3	0	10	0	7	103
exp	11	0	14	0	6	0	160
expf	5	1	5	1	2	1	218
log	18	0	12	0	5	0	227
logf	0	8	0	11	0	4	143
pow	32	0	31	0	21	0	338
powf	0	23	0	35	0	26	355
sin	8	0	11	0	6	0	109
sinf	0	3	0	9	0	5	97

Abbreviations Used

- ss or, e.g., cosf ≡ single precision
- sd or, e.g., cos ≡ double precision

Traditional `glibc` $^{energy}/_{instruction}$

Table 3: Parameters of traditional `glibc` implementations of floating point instructions.

Instruction	Area (um^2)	Freq. (MHz)	Energy (pJ)
`addss`	640	390	1
`addsd`	1500	390	2
`mulss`	6500	280	36
`mulsd`	16200	140	80

Evaluation Notes

- Evaluated in the NCSU 45nm predictive technology model
- For scale, one NN-based exp function uses 25 pJ
- Latency of one cycle

Traditional glibc $^{energy}/_{function}$

Table 4: Mean floating point energy

Function	Energy (pJ)
cos	967
cosf	365
exp	1158
expf	453
log	995
logf	415
pow	2561
powf	1292
sin	909
sinf	311

Observation

- Energy consumption is 2 orders of magnitude higher than NN-based implementation

NN Approximators for EDP Reductions

Table 5: NN-based EDP is significantly lower than glibc. Data is normalized to sin EDP, 3×10^{-19}.

	EDP in Multiples of sin EDP		
Func.	EDP-NN	EDP-Single	EDP-Double
cos	1	55	161
exp	4	1052	269
log	4	86	328
pow	31	666	1256
sin	1	44	144

Table 6: Applications that spend most of their cycles computing transcendental functions see large EDP improvements.

		Normalized EDP	
Benchmark	Transcendental Cycles	Single	Double
blackscholes	46%	56%	55%
swaptions	39%	62%	61%
bodytrack	2%	98%	98%
canneal	1%	99%	99%

Approximating Transcendental Functions

- Energy-delay product is 68x lower vs. glibc
- Mean squared error is 9×10^{-3}
- Application improvements follow Amdahl's law

NN-Based Accelerators in Applications – Accuracy

Table 7: Application output MSE and percent error using NN-based accelerators.

Benchmark	MSE $(\times 10^{-1})$	E[|%error|]
blackscholes	4.00	25%
bodytrack	2.00	30%
ferret	0.01	2%
swaptions	60.00	37%
canneal	2.89×10^8	0.0025%

MSE and Percent Error

- Qualitatively low error
- canneal has 1 large output, hence high MSE and low percent error

Library-Level NN-Based Accelerator Summary

Results

- Accelerators demonstrate EDP reductions...
  - 68x lower EDP than `glibc`
  - 78% of the EDP of traditional applications
- Library-level approximation is a suitable target for NN-based acceleration
- Work in this area can be improved by enabling NN-based accelerators to approximate additional functions and applications through...
  - Extensions to additional libraries
  - Capabilities to automatically identify and approximate functions

Appendix Contents

- Moore, C. (2011). Data Exascale-Class Computer Systems. Presented at *The Salishan Conference on High Speed Computing*.
- Venkatesh, G. et al. (2011). Qscores: trading dark silicon for scalable energy efficiency with quasi-specific cores. In *MICRO*.
- Esmaeilzadeh, H. et al. (2012). Neural acceleration for general-purpose approximate programs. In *MICRO*.
- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Math. Control Signal*, 2(4):303–314.
- Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.
- Chippa, V. K., Chakradhar, S. T., Roy, K., and Raghunathan, A. (2013). Analysis and characterization of inherent application resilience for approximate computing. In *DAC*.
- Volder, J. E. (1959). The cordic trigonometric computing technique. *IRE Tran. Comput.*, EC-8(3):330 –334.
- Walther, S. (1971). A unified algorithm for elementary functions. *AFIPS*.
- Chen, T., Chen, Y., Duranton, M., Guo, Q., Hashmi, A., Lipasti, M., Nere, A., Qiu, S., Sebag, M., and Temam, O. (2012). Benchnn: On the broad potential application scope of hardware neural network accelerators. In *IISWC*.
- Li, B., Shan, Y., Hu, M., Wang, Y., Chen, Y., and Yang, H. (2013). Memristor-based approximated computation. In *ISLPED*.