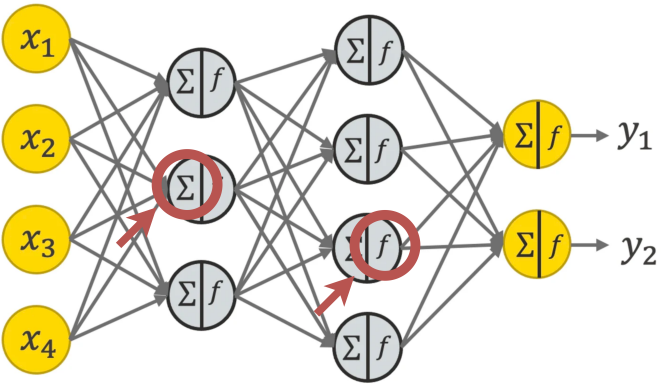
Semi-Automatic Compile-Time Math Intrinsic Optimization Using LLVM

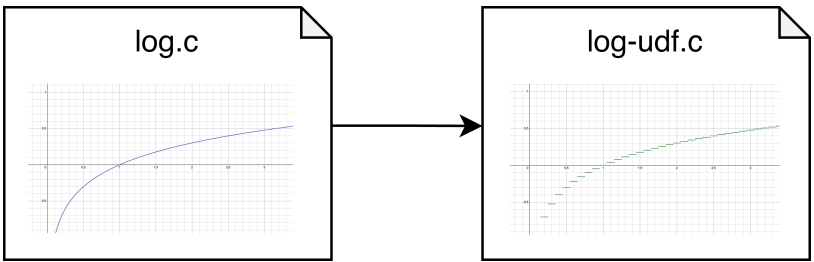
University of Texas at El Paso
Kristalys Ruiz Rohena
Christoph Lauter
Shirley Moore

- Numerical analysis, transcendental functions, and number representations in AI.
- Manipulation of implementations within models.
- Tools: **Herbie**[1], **STOKE**[2], **Precimonious** [3], **MPTorch** [4], [5]



Problem

- Natively incorporate **any** UDF into Torch Library.
- AI-specific dynamic instrumentation and monitoring.
- Desirable features:
  - Faster than compiling the entire library.
  - Replicable across architectures.

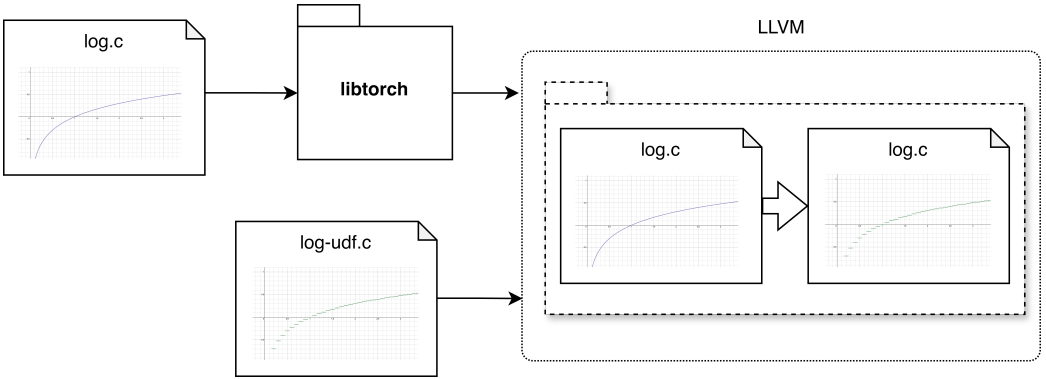


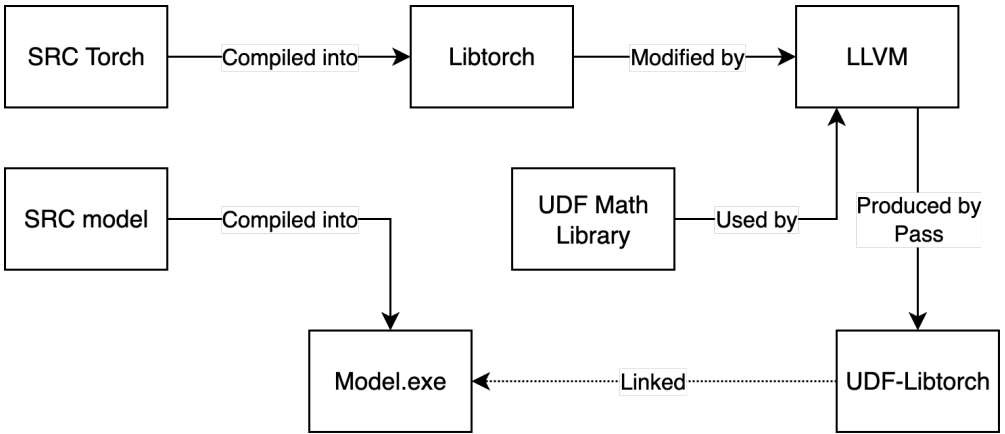
- **Correctly Rounded**: A result rounded to the nearest representable value.
- **LLVM** (Low-Level Virtual Machine): A compiler framework and tool that can optimize code during compilation and at runtime.
- **UDF** (User-Defined Function): A function created by the user to extend capabilities.
- **LibTorch**: PyTorch Backend library written in C++.

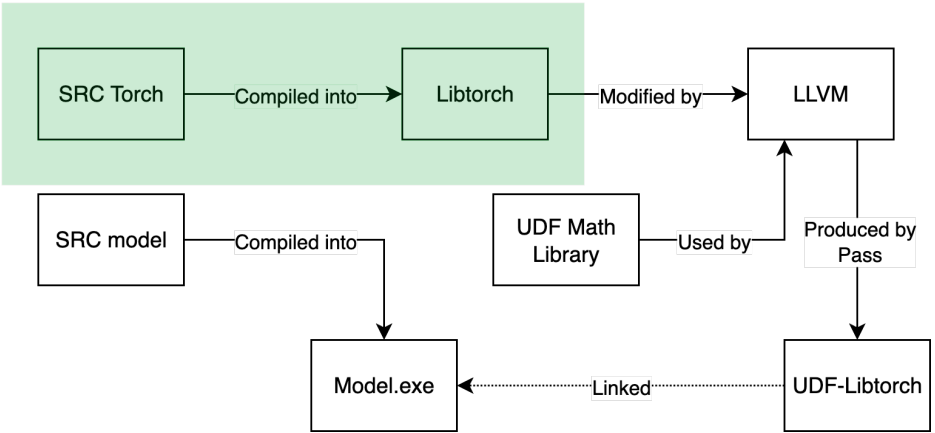
Example

```
Tensor& log_udf_out(const Tensor& self, Tensor& out) {
    float min_val, max_val;
    compute_minmax(self, min_val, max_val);
    ProfilerWriter writer("profiled.bin");
    ProfileEvent e;
    auto self_ptr = self.data_ptr<float>();
    auto out_ptr = out.data_ptr<float>();
    int64_t n = self.numel();
    e.timestamp_start = get_timestamp();
    for (int64_t i = 0; i < n; i++) {
        out_ptr[i] = std::log(self_ptr[i]);
    }
    e.timestamp_end = get_timestamp();
    e.op_id = OP_LOG;
    e.min_val = min_val;
    e.max_val = max_val;
    writer.logEvent(e);
    return out;
}
```

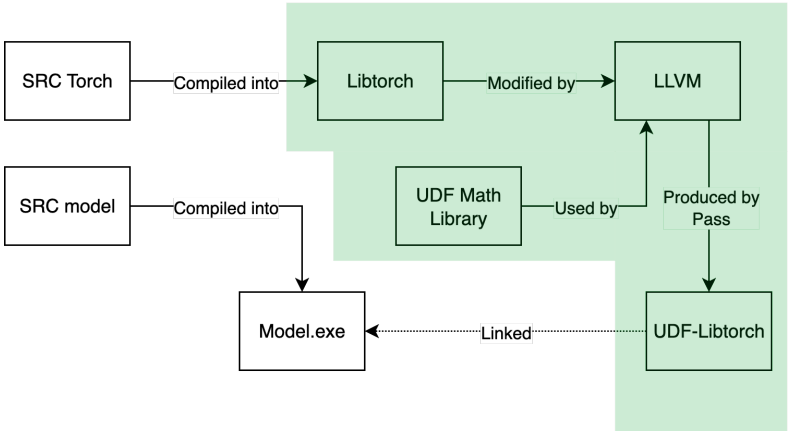
System Overview

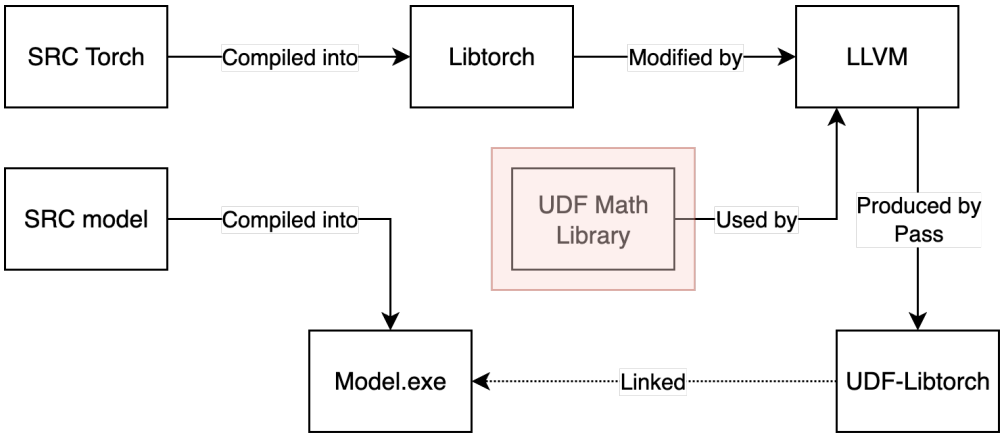



SRC Torch
Compiled into
Libtorch
Modified by
LLVM
SRC model
Compiled into
UDF Math Library
Used by
Produced by Pass
Model.exe
Linked
UDF-Libtorch

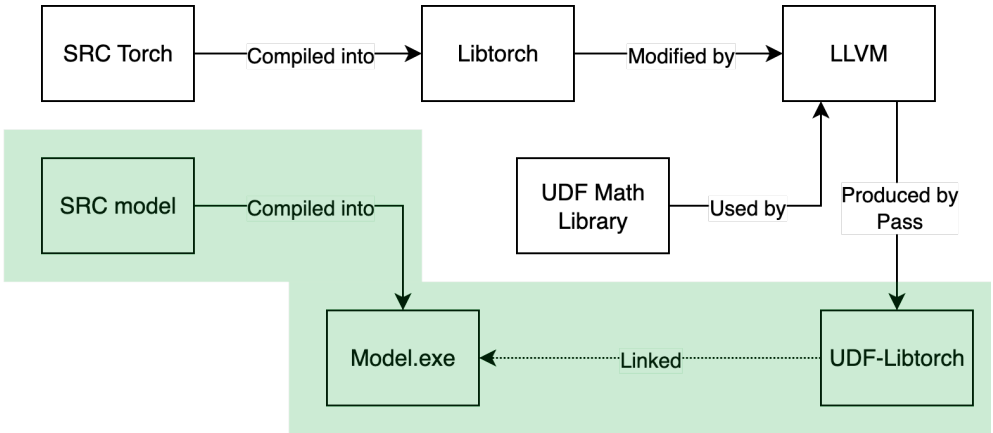

SRC Torch
Compiled into
Libtorch
Modified by
LLVM
SRC model
Compiled into
UDF Math Library
Used by
Produced by Pass
Model.exe
Linked
UDF-Libtorch

LLVM Pass


SRC Torch
Compiled into
Libtorch
Modified by
LLVM
SRC model
Compiled into
UDF Math Library
Used by
Produced by Pass
Model.exe
Linked
UDF-Libtorch


SRC Torch
Compiled into
Libtorch
Modified by
LLVM
SRC model
Compiled into
UDF Math Library
Used by
Produced by Pass
Model.exe
Linked
UDF-Libtorch

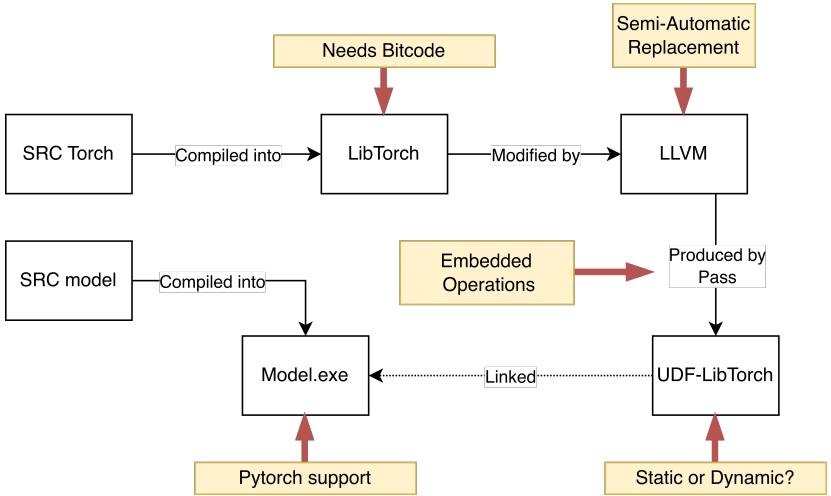
Model Execution

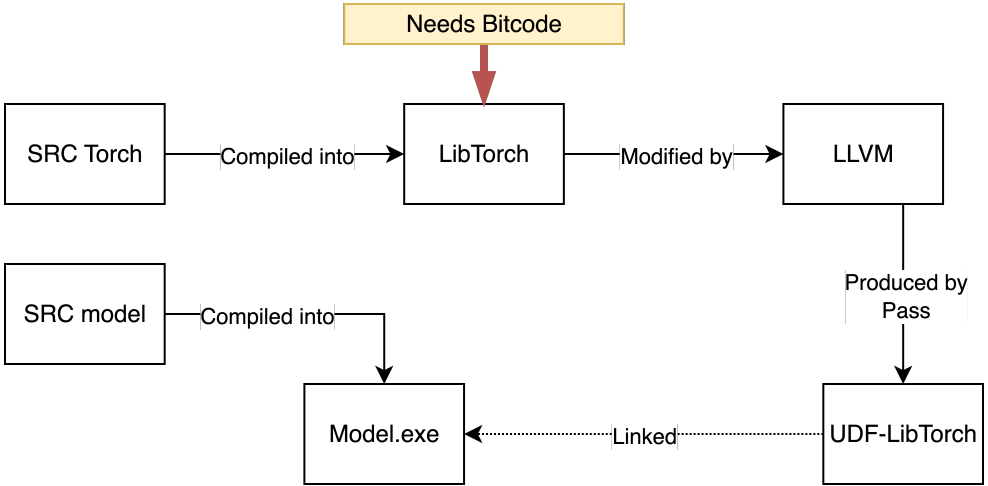


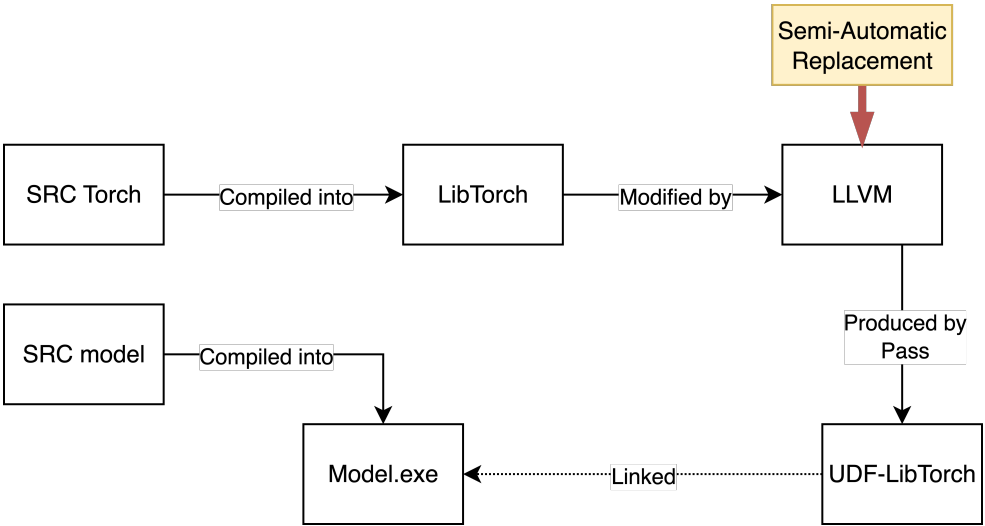
The LLVM pass automates the process of substituting the original implementations with the UDF versions.

- *Optional:* Modify the ATen library to allow LLVM passes to access and manipulate operations.
- Compile static libraries with bitcode (LLVM intermediate representation).
- Compile user-defined functions (UDFs) for use in the LLVM pass.
- **Replace function bodies while keeping declarations to preserve references.**
- *Optional:* Convert static libraries to dynamic libraries.
- **The modified UDF-LibTorch library can be used interchangeably with the original LibTorch build.**

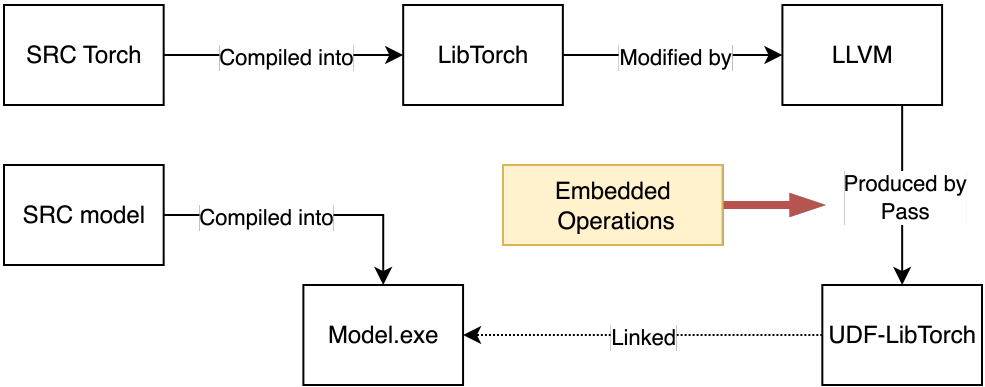
Challenges


Needs Bitcode
Semi-Automatic Replacement
SRC Torch
Compiled into
LibTorch
Modified by
LLVM
SRC model
Compiled into
Embedded Operations
Produced by Pass
Model.exe
Linked
UDF-LibTorch
Pytorch support
Static or Dynamic?

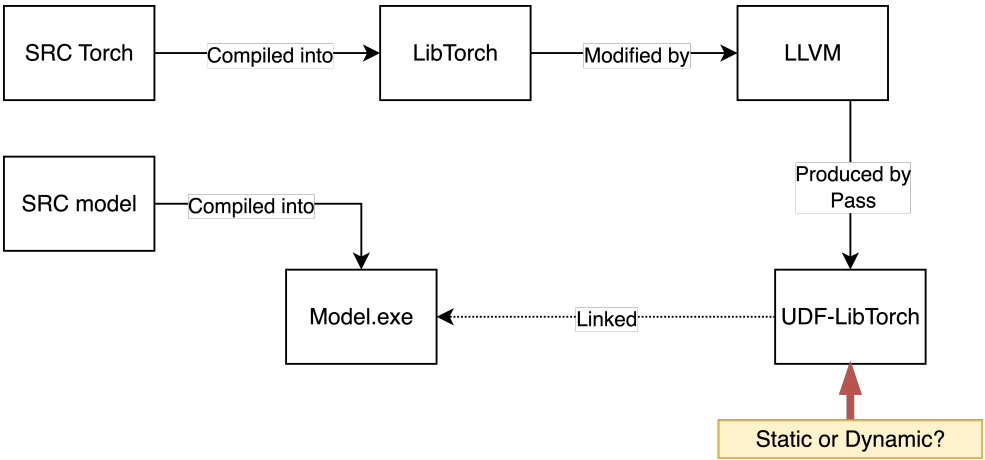

Needs Bitcode
SRC Torch
Compiled into
LibTorch
Modified by
LLVM
SRC model
Compiled into
Produced by
Pass
Model.exe
Linked
UDF-LibTorch


Semi-Automatic Replacement
SRC Torch
Compiled into
LibTorch
Modified by
LLVM
SRC model
Compiled into
Produced by Pass
Model.exe
Linked
UDF-LibTorch

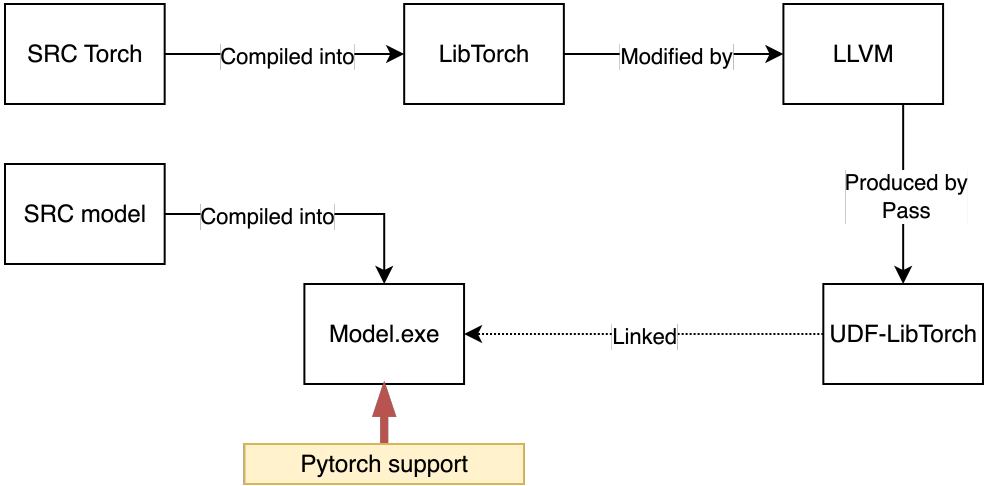
Optimized Operations



Linking


SRC Torch
Compiled into
LibTorch
Modified by
LLVM
SRC model
Compiled into
Produced by Pass
Model.exe
Linked
UDF-LibTorch
Static or Dynamic?

Platform


SRC Torch
Compiled into
LibTorch
Modified by
LLVM
Produced by
Pass
SRC model
Compiled into
Model.exe
Linked
UDF-LibTorch
Pytorch support

- Objectives:
  - Perform an initial profiling of key transcendental functions: $\exp, \log, \sin, \cos$.
  - Assess the impact of correct rounding on accuracy.
- Experimental Setup:
  - **Dataset:** MNIST, CIFAR10
  - **Architectures:** LeNet (CNN), ResNet
  - Activation Function ELU[6]:

$$\mathrm{ELU}(x) = \begin{cases} x, & \text{if } x > 0, \\ \alpha\left(e^x - 1\right), & \text{if } x \leq 0. \end{cases}$$

ResNet + Cifar10

Operation	Count	Library	Min	Max
EXP	122,225	STD	-58.4268	0
		Core-Math	-74.1376	0

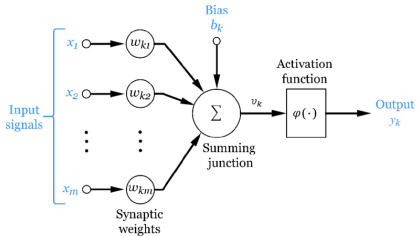
MNIST + LeNet

Operation	Count	Library	Min	Max
LOG	12,100	STD	1	9.24225
		Core-Math	1	9.24225
EXP	48,400	STD	-78.1515	0
		Core-Math	-79.0715	0

**With correctly rounded library CORE-Math[7], accuracy remained the same.*

Assumptions:

- The activation function has a transcendental function.
- Optimizer: Stochastic Gradient Descent.
- For the error propagation, the average is evaluated in fully connected layers.



Layer	Details	Input	Output
Conv(1, 10, 5)	Has bias	1×28×28	10×24×24
MaxPool2d(2)		10×24×24	10×12×12
ELU		10×12×12	10×12×12
Conv(10, 20, 5)	Has bias	10×12×12	20×8×8
Flatten		20×4×4	1×320
ELU		1×320	1×320
FCL(320, 50)	Has bias	1×320	1×50
ELU		1×50	1×50
FCL(50, 10)	Has bias	1×50	1×10
LogSoftmax		1×10	1×10

Operation Ratio per Stage

Stage	fma	div	mult	add	exp & log
Forward			130	130	1
Back Propagation	5	5	12000	6700	1

- Profiling takes at most 1% of runtime.
- Transcendental functions account for less than 4% of runtime with correctly rounded (CR) and less than 1% with the standard library (STD).

- Developed a tool that enables instrumentation in Torch workloads.
- **Future Work:**
  - Evaluate and define error tolerance.
  - Integrate the UDF-LibTorch library into the PyTorch API.
  - Develop a transcendental function library specifically for CUDA.
  - Explore transformer-based models.

Thanks!

*This material is based upon work supported by the NSF under Grant Number #2311708. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the NSF.

[1] P. Panchekh, A. Sanchez-Stern, J. R. Wilcox, and Z. Tatlock, "Automatically Improving Accuracy for Floating Point Expressions,", 2015. DOI: o.oi.rogr/1g0/1.101.4151/4nn5n/2n7nn3n7.9nn2n4n.2nn7n37959.

[2] E. Schkufza, R. Sharma, and A. Aiken, "Stochastic superoptimization," *ACM SIGARCH Computer Architecture News*, vol. 41, no. 1, pp. 305–316, Mar. 2013, ISSN: 0163-5964. DOI: 10.1145/2490301.2451150.

[3] C. Rubio-González, C. Nguyen, H. D. Nguyen, *et al.*, "Precimonious: Tuning assistant for floating-point precision,", IEEE, 2025. DOI: .org/10.1145/2503210.2503296..

[4] M. Contributors, *Mptorch: Mixed precision simulation framework for pytorch*, https://github.com/mptorch/mptorch, GitHub repository, accessed 2025-11-02, 2024. [Online]. Available: https://github.com/mptorch/mptorch.

[5] Q. Contributors, *Qpytorch: A low-precision arithmetic simulation framework*, https://github.com/Tiiiger/QPyTorch, GitHub repository, accessed 2025-11-02, 2020. [Online]. Available: https://github.com/Tiiiger/QPyTorch.

[6] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, *Fast and accurate deep network learning by exponential linear units (elus)*, 2016. DOI: 10.48550/arxiv.1511.07289. arXiv: 1511.07289 [cs.LG].

[7] A. Sibidanov, P. Zimmermann, and S. Glondu, "The core-math project," in *2022 IEEE 29th Symposium on Computer Arithmetic (ARITH)*, 2022, pp. 26–34. DOI: 10.1109/ARITH54963.2022.00014.

[8] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, IEEE, 2009, pp. 248–255.

[9] J. Schaap, S. Dennis, Sprokholt, P. Soham, and Chakraborty, "Evaluating Stochastic Floating-Point Superoptimization with STOKE,", 2022.

[10] K. Jia and M. Rinard, *Exploiting verified neural networks via floating point numerical error*, 2021. arXiv: 2003.03021 [cs.LG]. [Online]. Available: https://arxiv.org/abs/2003.03021.

[11] M. Hardt, MRTZ@GOOGLE.COM, B. Recht, BRECHT@BERKELEY.EDU, Y. Singer, and SINGER@GOOGLE.COM, "Train faster, generalize better: Stability of stochastic gradient descent,", 2016.

[12] O. Kupriianova and C. Lauter, "Metalibm: A mathematical functions code generator," in *Mathematical Software – ICMS 2014*, H. Hong and C. Yap, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 713–717, ISBN: 978-3-662-44199-2.

[13] T. Ribeiro and F. H. Oliveira, "Desenvolvimento de modelos de previsão de coeficiente de atrito em pistas de pouso e decolagem brasileiras com redes neurais artificiais," *TRANSPORTES*, vol. 31, e2792, Aug. 2023. DOI: 10.58922/transportes.v31i2.2792.

[14] K. Melcher, *A friendly introduction to [deep] neural networks*, [Online]. Available: https://www.knime.com/blog/a-friendly-introduction-to-deep-neural-networks.