Training Neural Networks on Embedded Devices

Targeting Embedded Environments

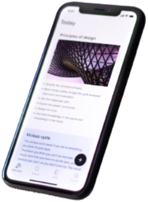
Prasanth Shaji, Deepak Venkataram

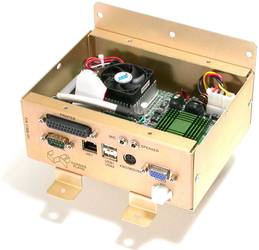
Master's Thesis

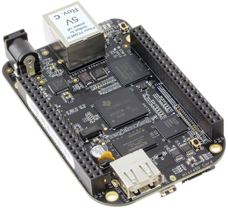
Uppsala University

Outline

Embedded Devices







Neural Network Applications on Embedded Devices

UPPSALA UNIVERSITET

Goal : Neural Network Training on Scania ECU

Goal : Neural Network Training on Scania ECU

Predictive vehicle maintenance using data from a fleet of connected trucks

Why Training on the ECU?

- Training on devices reduces network usage, ensures data privacy, etc.
- Repurpose existing ECUs to perform neural network related tasks

Scope *Repurpose Scania ECU, train neural network model, assess training performance*

Goal : Neural Network Training on Scania ECU

Predictive vehicle maintenance using data from a fleet of connected trucks

Why Training on the ECU?

- Training on devices reduces network usage, ensures data privacy, etc.
- Repurpose existing ECUs to perform neural network related tasks

Scope *Repurpose Scania ECU, train neural network model, assess training performance*

Goal : Neural Network Training on Scania ECU

UPPSALA UNIVERSITET

Predictive vehicle maintenance using data from a fleet of connected trucks

Why Training on the ECU?

- Training on devices reduces network usage, ensures data privacy, etc.
- Repurpose existing ECUs to perform neural network related tasks

Scope *Repurpose Scania ECU, train neural network model, assess training performance*

Goal : Neural Network Training on Scania ECU

UPPSALA UNIVERSITET

Predictive vehicle maintenance using data from a fleet of connected trucks

Why Training on the ECU?

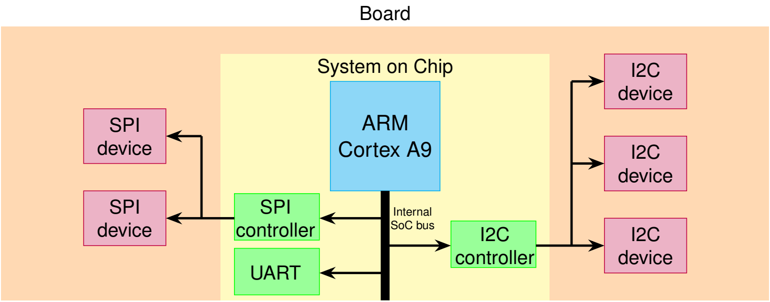
- Training on devices reduces network usage, ensures data privacy, etc.
- Repurpose existing ECUs to perform neural network related tasks

Scope *Repurpose Scania ECU, train neural network model, assess training performance*

UPPSALA UNIVERSITET

Embedded Devices : Hardware Outline

ARM | Silicon Vendor | System Maker



ARM Boards : Examples

SoC	Application Domain	Architecture	Processor Core
nRF51822	Ultra low power, BLE	ARM v6-M	ARM Cortex M0
AM2732	Automotive	ARM v7-R	ARM Cortex R5
AM3358	Industrial / IoT	ARM v7-A	ARM Cortex A8
i.MX6S	Multimedia applications	ARM v7-A	ARM Cortex A9
BCM2711	Educational / IoT	ARM v8-A	ARM Cortex-A72
Snapdragon 460	Smartphones	ARM v8-A	ARM Cortex-A73, A53
Google Tensor	Pixel 6	ARM v8.2-A	ARM Cortex-X1, A76, A55
Apple A16 Bionic	iPhone 14 Pro	ARM v8.6-A	APL1W10
Apple M1	Macbooks / Mac Mini / iPads	ARM v8.5-A	APL1102

Embedded Linux

Operating System for Embedded Devices
- Customized for Specific Hardware

Kernel, Bootloaders, Device Trees
Cross Compiling Toolchains, Root Filesystems

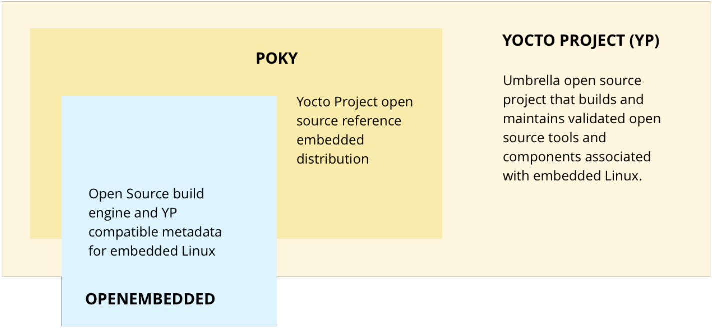
Embedded Linux

Operating System for Embedded Devices
- Customized for Specific Hardware

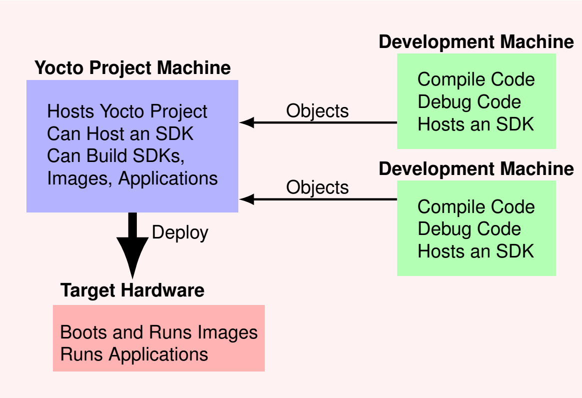
Kernel, Bootloaders, Device Trees
Cross Compiling Toolchains, Root Filesystems

Build Systems

Yocto Project

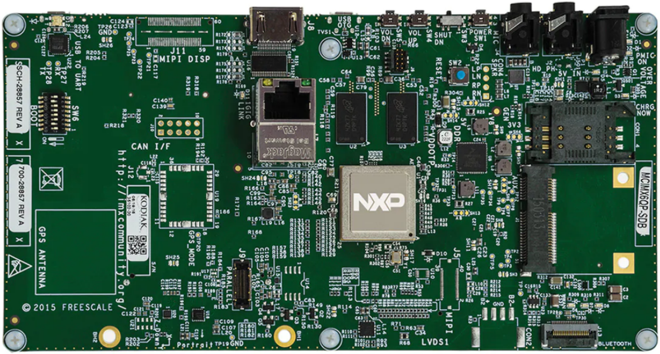


Development Setup

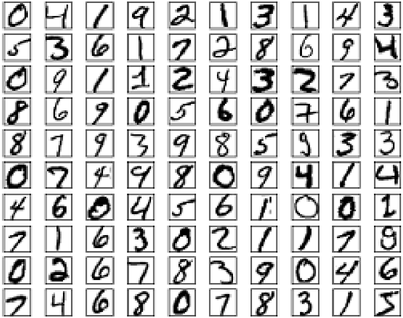


Software Versions

Scania ECU to MCIMX6Q-SDB

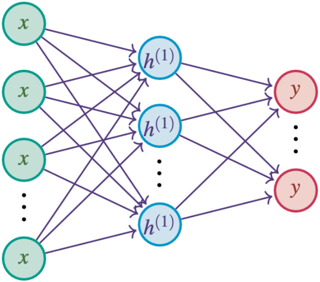


Handwritten Digit Recognition Neural Network



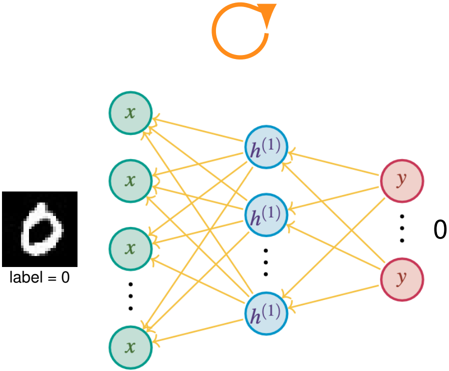
MNIST

Neural Network



$$y_k = \sigma\left(\sum_{j=0}^{m} w_{kj} x_j\right)$$

Neural Network Training



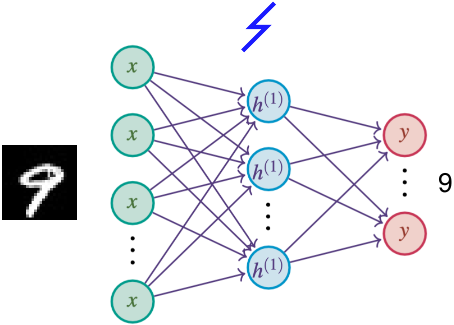
Learning Algorithm

Algorithm 1 Mini Batch Gradient Descent with learning rate η and the Mean Squared Error (`MSE`) cost function

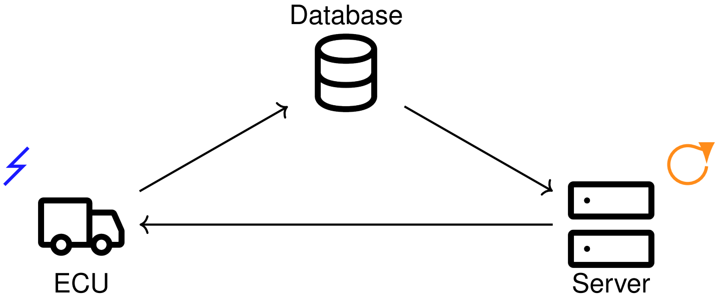
Require: initial weights $w^{(0)}$, number of epochs E, batch size B, training data with T entries
Ensure: final weights $w^{(E*T)}$
  for $e = 0 \to E-1$ **do**
    for $b = 0 \to T/B$ **do**
      for $t = b*B \to (b+1)*B$ **do**
        estimate $\nabla\mathscr{L}(w^{(t)})$ ▷ $\mathscr{L}$ here is `MSE`
        compute $\Delta w^{(b)} + = -\nabla\mathscr{L}(w^{(t)})$
      end for
      $w^{(e+1)} := w^{(e)} + \eta\Delta w^{(e)}$
    end for
  end for
  return $w^{(T)}$

UPPSALA UNIVERSITET

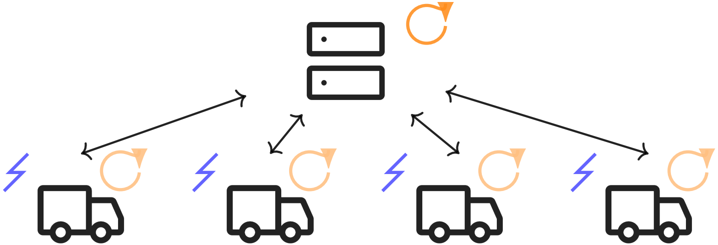
Neural Network Inference



Traditional Paradigm



Federated Learning



HDR-NN Implementations

github.com/paperbound/hdrnn-arm

- `C` based HDR-NN
- `C++` / `Eigen`
- `C++` / `libtorch` (PyTorch)
- `Python` / `Numpy`

C based HDR-NN


```c
/* HDR Neural Network */
typedef struct
{
    float bias;
    float *weights;
    float *nabla_w;
} Neuron;

typedef struct LayerT
{
    int size;
    int incidents;
    Neuron *neurons;
    float *activations;
    float *z_values;
    float *nabla_b;
    struct LayerT *next;
    struct LayerT *previous;
} Layer; // Network layers except for input

typedef struct
{
    Layer *layers;
    int depth;
} Network; // HDRNN
```

```c
static void feed_forward(Network *network, float *image)
{
    // First layer is the image
    float *activations = image;

    Layer *layer = network->layers;
    while (layer != NULL)
    {
        for (int i = 0; i < layer->size; i++)
        {
            Neuron *neuron = &layer->neurons[i];
            float zvalue = 0;
            for (int j = 0; j < layer->incidents; j++)
            {
                zvalue += neuron->weights[j] * activations[j];
            }
            layer->z_values[i] = zvalue + neuron->bias;
            layer->activations[i] = sigmoid(layer->z_values[i]);
        }
        activations = layer->activations;
        layer = layer->next;
    }
}
```

C++ / Eigen based HDR-NN

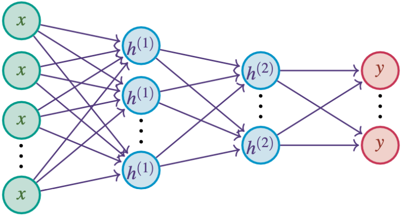
- `Eigen` is a `C++` template library for linear algebra
- Analogous to `Python` / `Numpy`

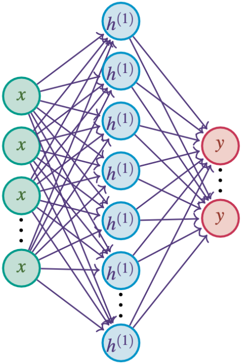
C++ / libtorch based HDR-NN

- PyTorch doesn't support MCIMX6Q
- Porting `libtorch` using QEMU

UPPSALA UNIVERSITET

HDR-NN configurability





HDR-NN configurability

```
→ ./bin/hdrnn
nothing to do
Usage: ./bin/hdrnn

        <command> [<args>]

Commands:
        infer [-i, --image IMAGE_PATH] [-n, --net c-math.nn]
        train [-s, --shape 32] [-e, --epochs 30]
         [-q, --quiet] [-bs, --batch_size=10]
         [-lr, --learning_rate 3] [-n, --net c-math.nn]
```

Experiment Considerations

- Try multiple shapes m
- Compare Accuracies
- Compare Execution Time

UPPSALA UNIVERSITET

Introduction

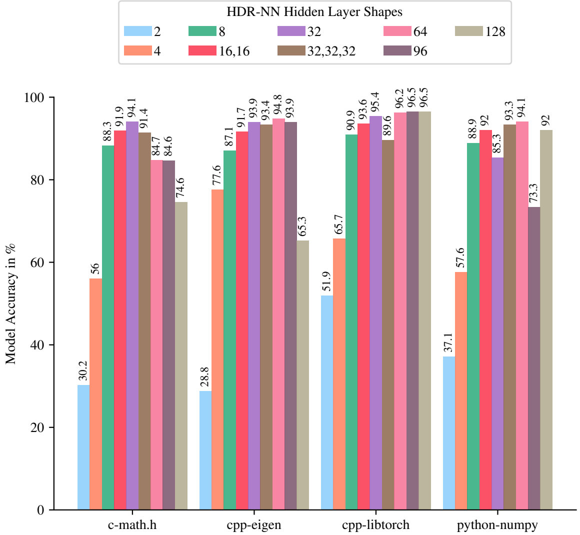
Embedded Linux

Neural Networks

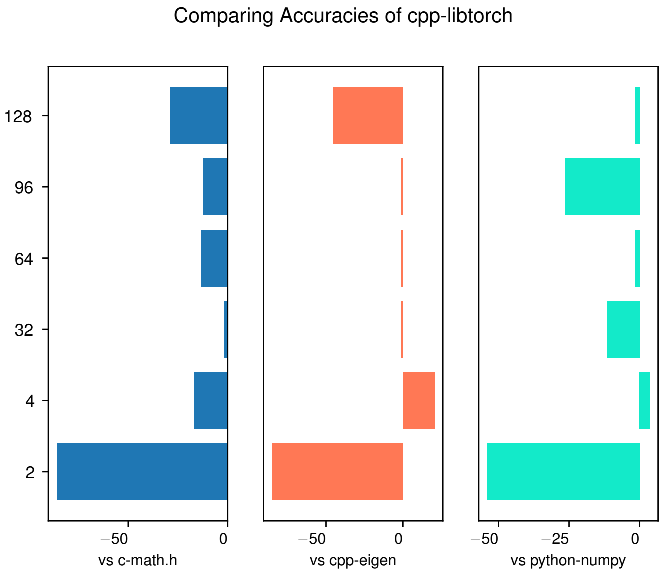
Benchmark Applications

Discussion

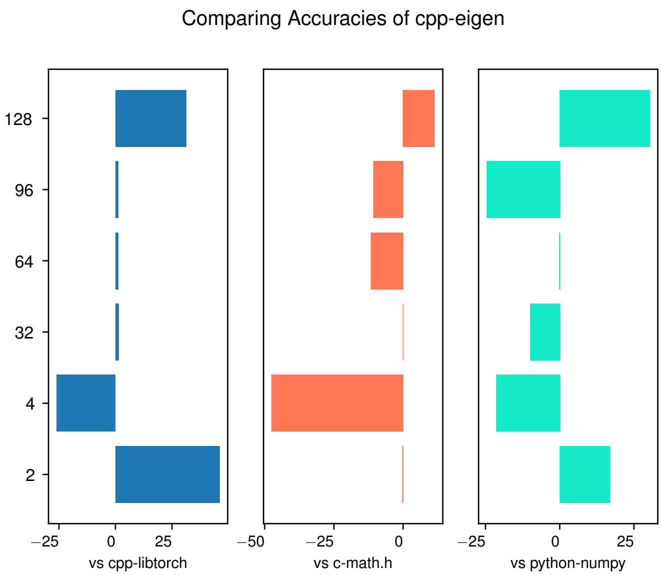
Accuracy



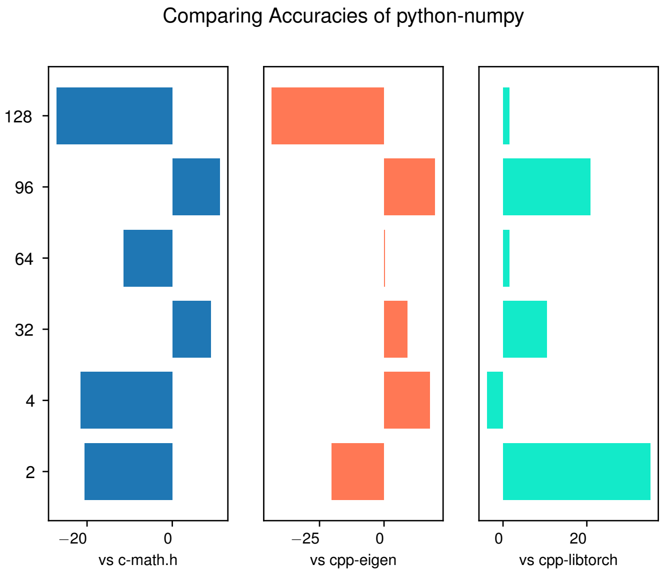
UPPSALA UNIVERSITET

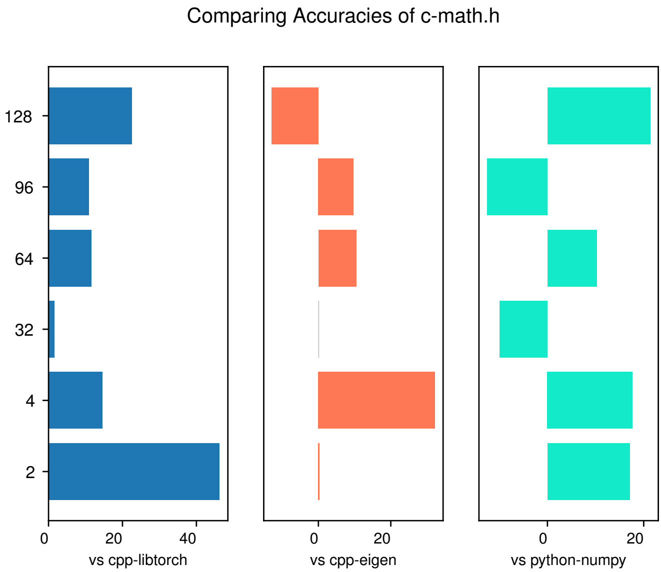


UPPSALA UNIVERSITET



UPPSALA UNIVERSITET

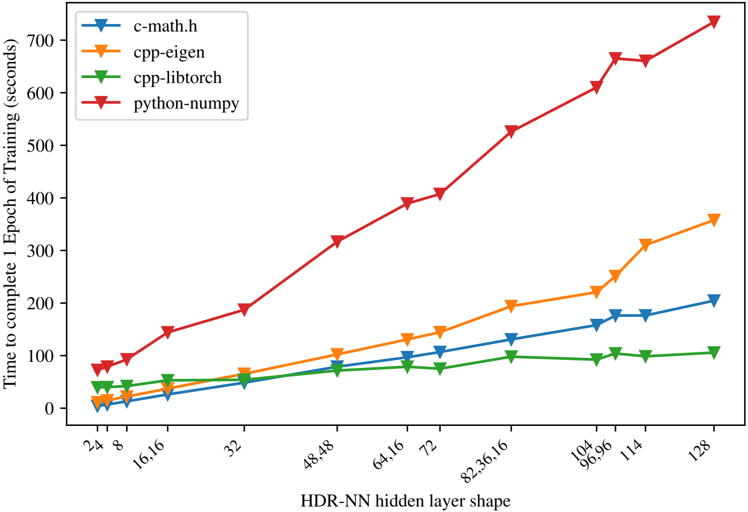



Comparing Accuracies of c-math.h
128
96
64
32
4
2
0
20
40
vs cpp-libtorch
0
20
vs cpp-eigen
0
20
vs python-numpy

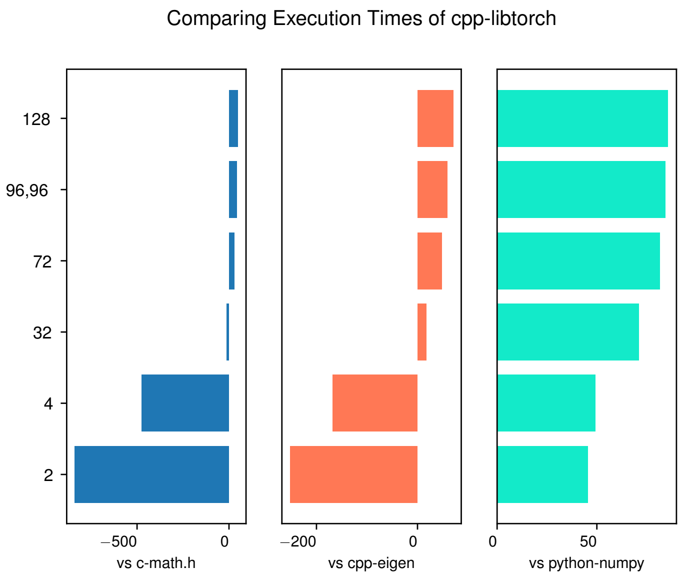
UPPSALA UNIVERSITET

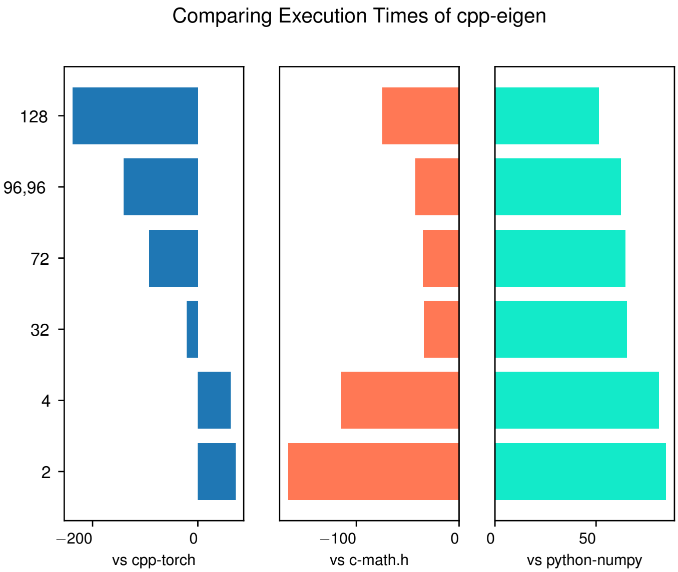
- Introduction
- Embedded Linux
- Neural Networks
- **Benchmark Applications**
- Discussion

Training Time

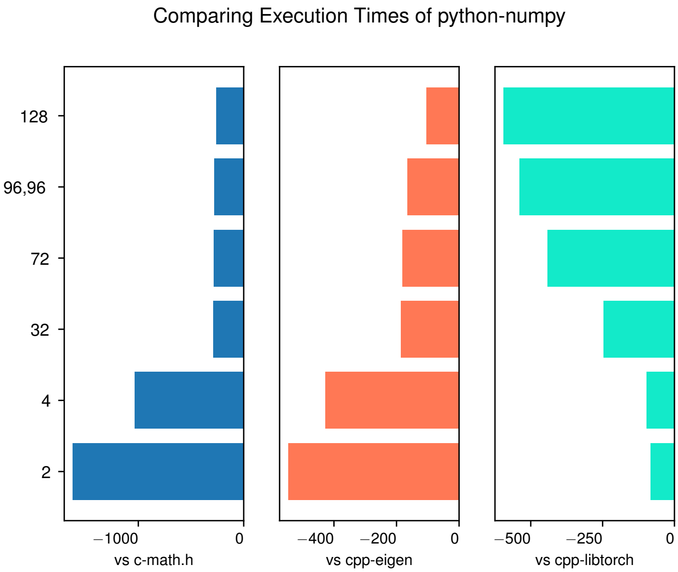


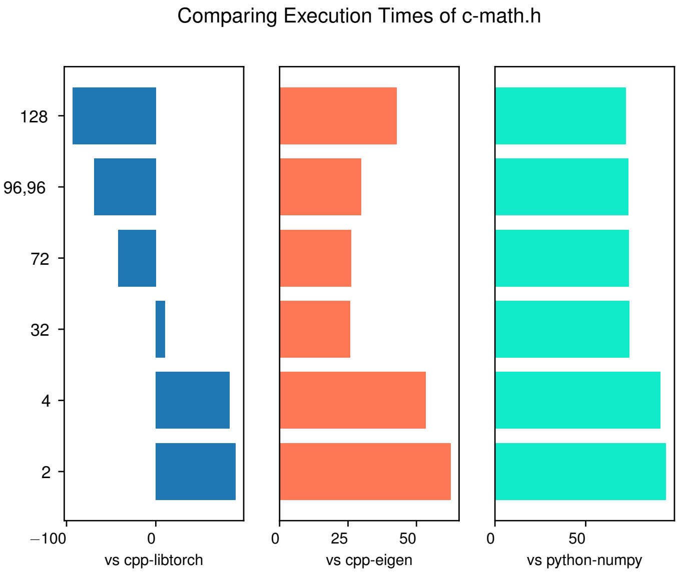
UPPSALA UNIVERSITET




Comparing Execution Times of cpp-eigen
128
96,96
72
32
4
2
−200
0
vs cpp-torch
−100
0
vs c-math.h
0
50
vs python-numpy

UPPSALA UNIVERSITET




Comparing Execution Times of c-math.h
128
96,96
72
32
4
2
−100
0
vs cpp-libtorch
0
25
50
vs cpp-eigen
0
50
vs python-numpy

UPPSALA UNIVERSITET

Reverse Engineering Scania ECU

- Goal : Get a vanilla U-Boot running on ECU
- Extract device tree information
- Silent bootloader, No visibility into Board/SoM fork

Neural Network Software Development on Embedded Devices

- Embedded Platforms are fragmented in both *hardware* and *software*
- C is easier to target harder to Implement
- Python commonly found in Neural Network Research, harder to port

Future Work

- Reverse Engineering Efforts on the ECU
- Performance engineering the `C` based implementation
- Adding more target devices
- Adding more implementations (Tensorflow, TTE)

End

Thank you