Training Neural Networks on Embedded Devices

Targeting Embedded Environments

Prasanth Shaji, Deepak Venkataram

Master's Thesis

Uppsala University

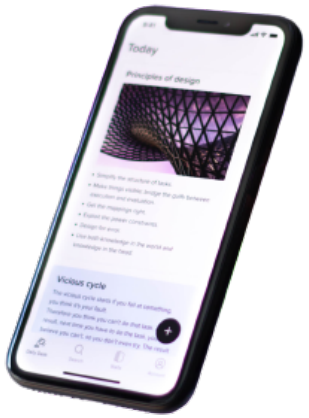
August 21, 2023

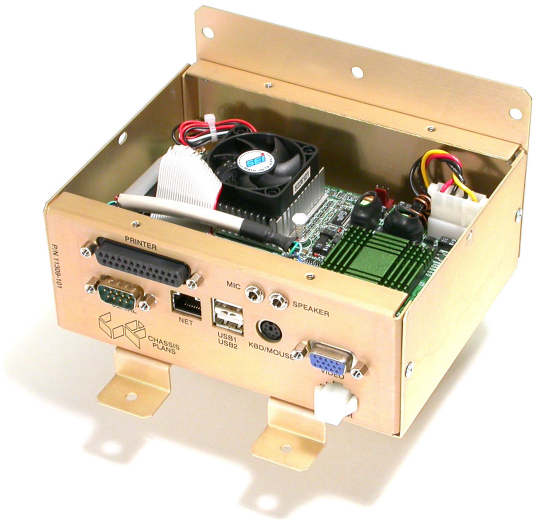
UPPSALA UNIVERSITET

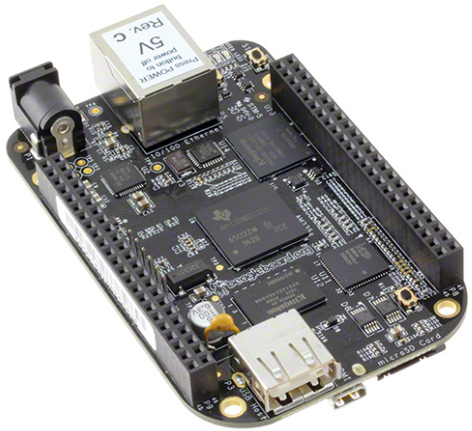
Outline

Neural Network Applications on Embedded Devices

Embedded Devices







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?

- Repurpose existing ECU units to perform neural network related tasks
- Training on devices to reduce network bandwidth usage, ensure data privacy, etc.

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?

- Repurpose existing ECU units to perform neural network related tasks
- Training on devices to reduce network bandwidth usage, ensure data privacy, etc.

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?

- Repurpose existing ECU units to perform neural network related tasks
- Training on devices to reduce network bandwidth usage, ensure data privacy, etc.

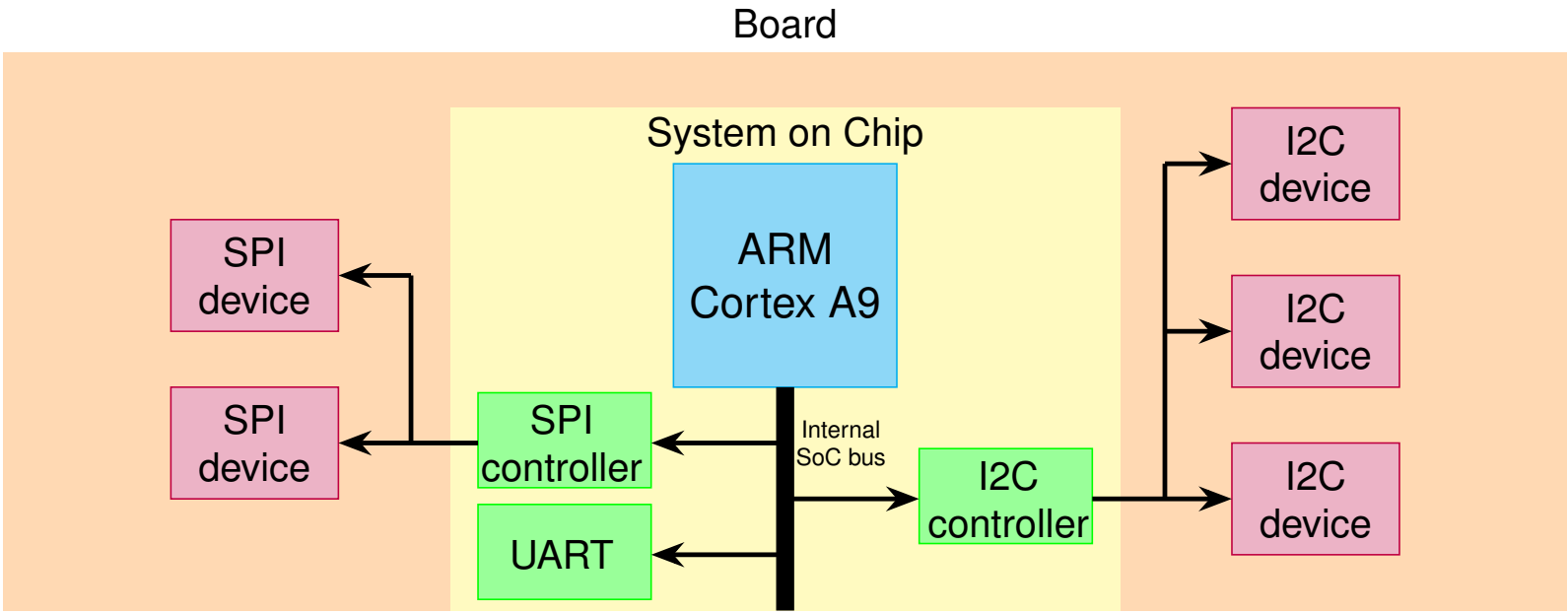
Scope *Repurpose Scania ECU, train neural network model, assess training performance*

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 - Kryo 240	Smartphones	ARM v8-A	ARM Cortex-A73, ARM Cortex-A53
Apple A16 Bionic	iPhone 14 Pro	ARM v8.6-A	APL1W10
Apple M1	Macbooks / Mac Mini / iPads	ARM v8.5-A	APL1102

ARM Boards : Outline

ARM | Silicon Vendor | System Maker



Embedded Linux

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

Operating System for Embedded Devices
- Customized for Specific Hardware

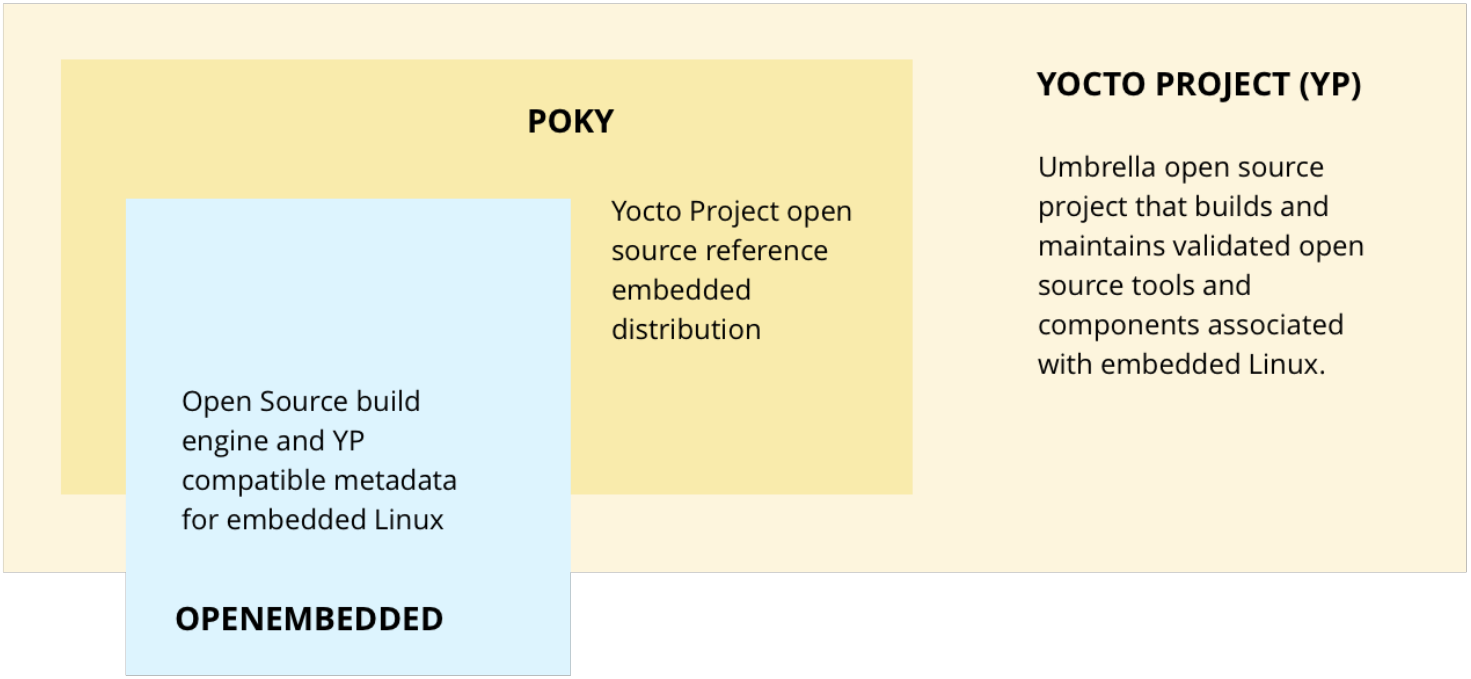
Embedded Linux

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

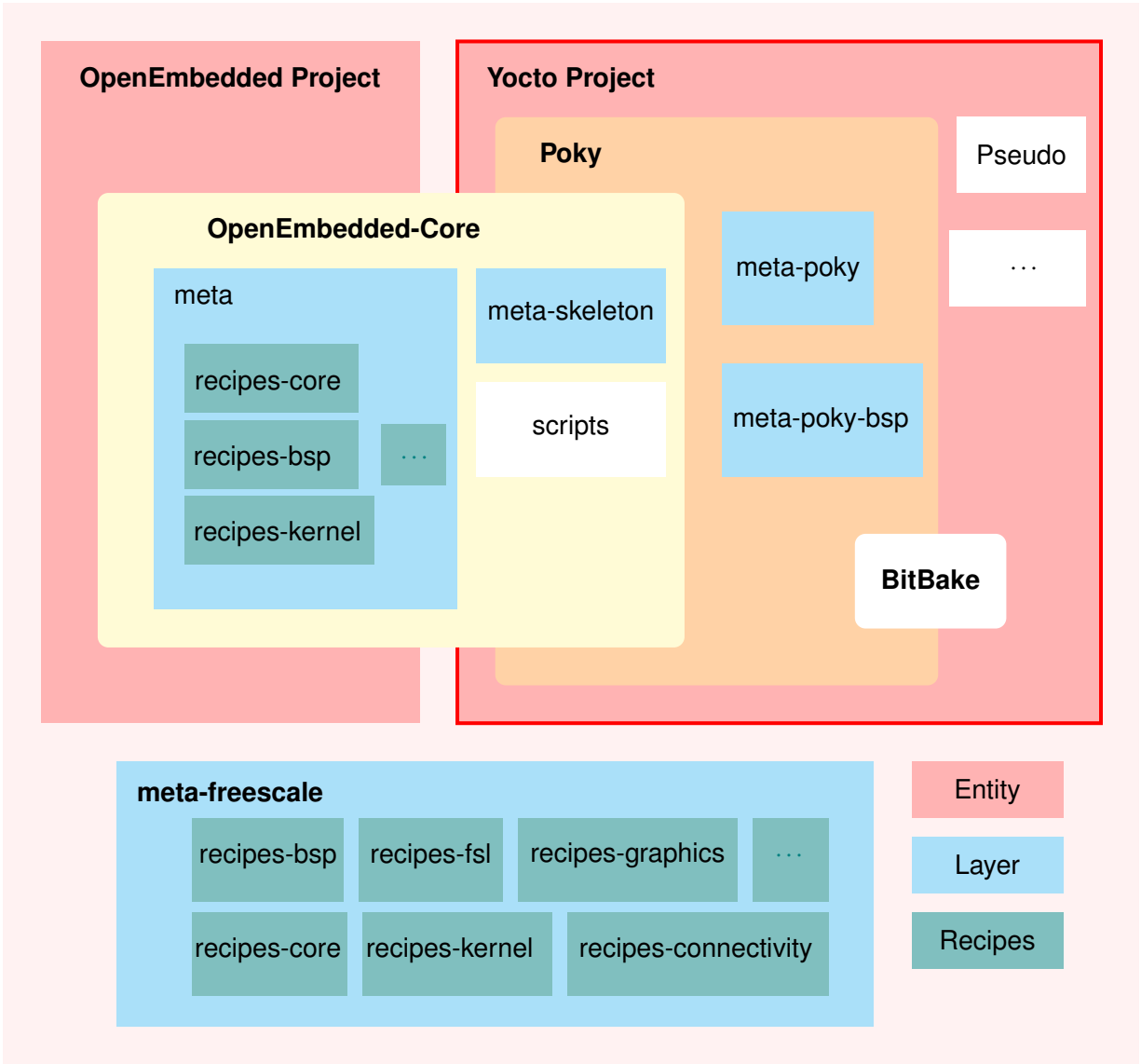
Operating System for Embedded Devices
- Customized for Specific Hardware

Build Systems

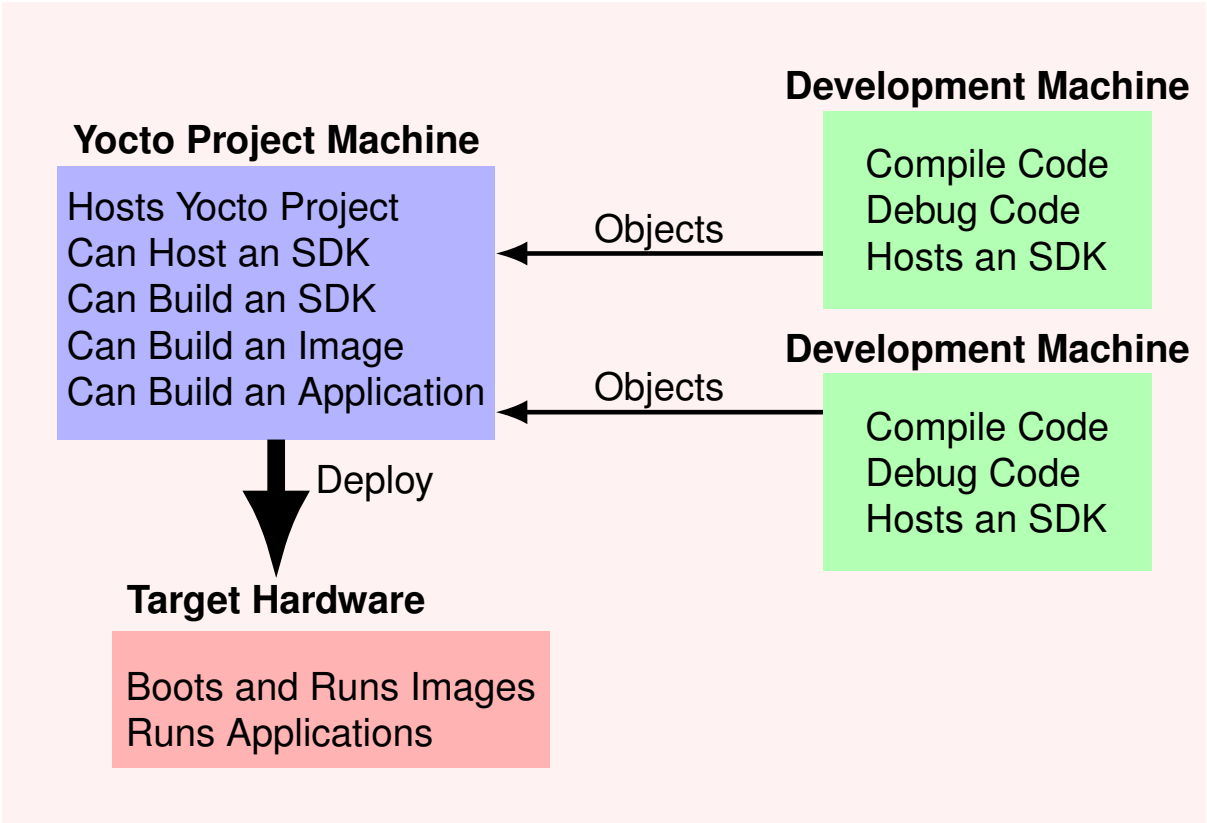
Yocto Project



The Yocto Project

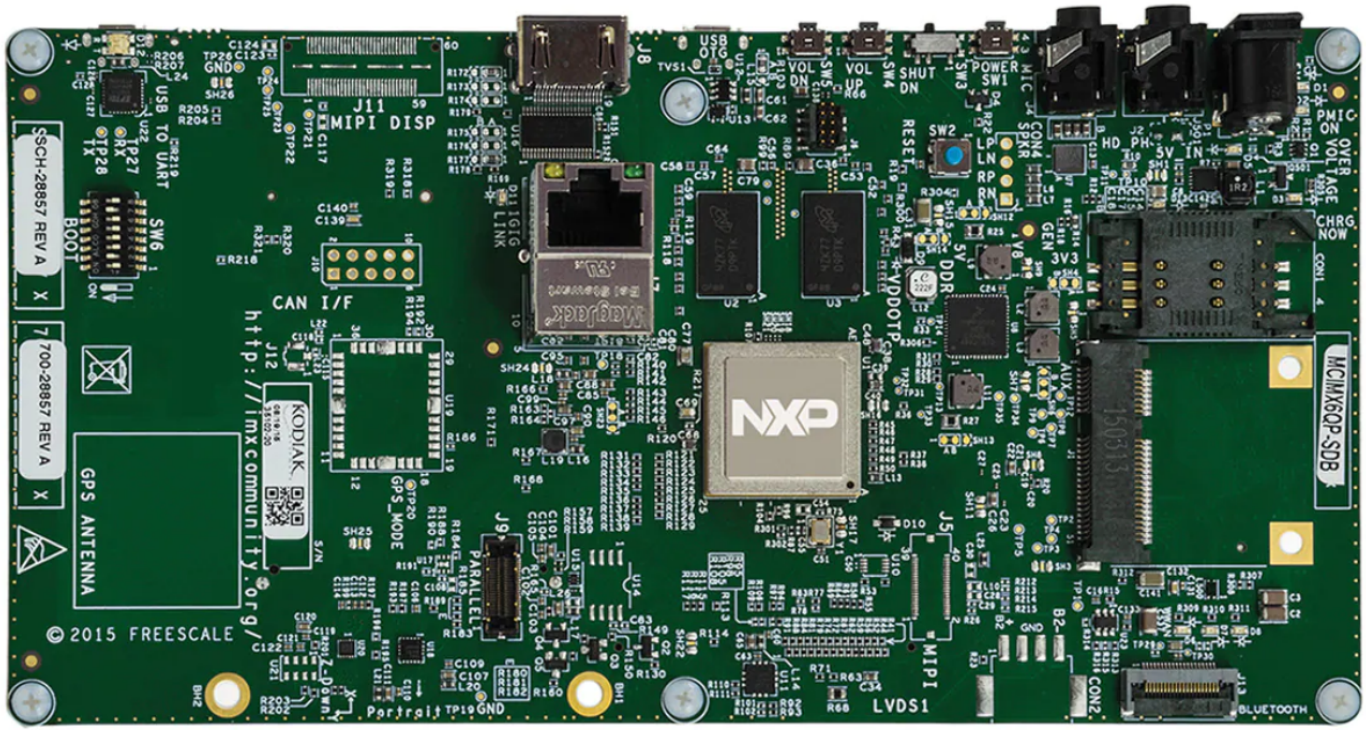


Yocto Project : Development

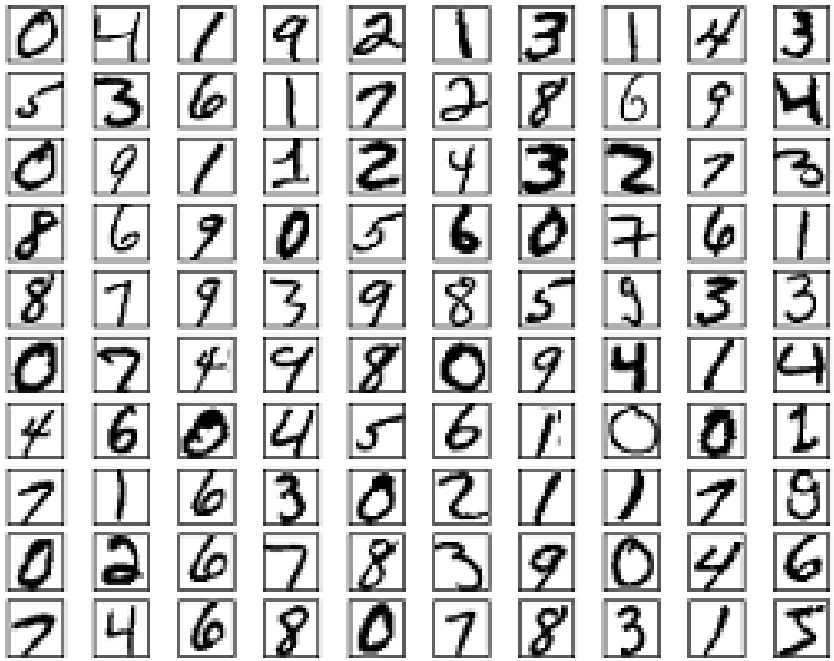


Software Versions

Scania ECU to MCIMX6Q-SDB

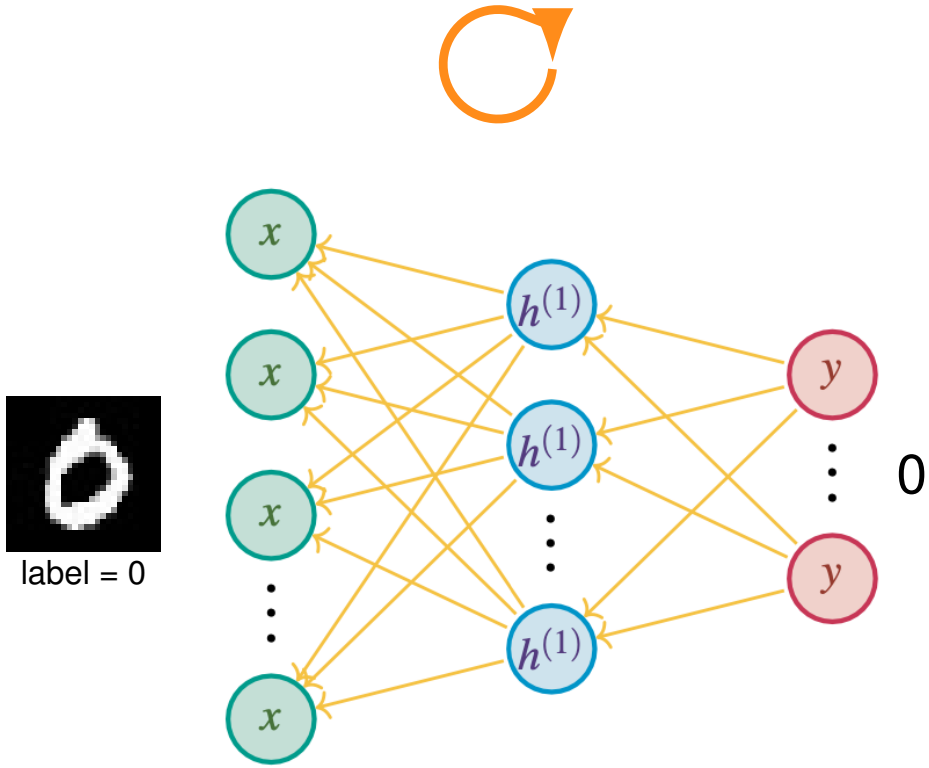


Handwritten Digit Recognition Neural Network



MNIST

Neural Network Training



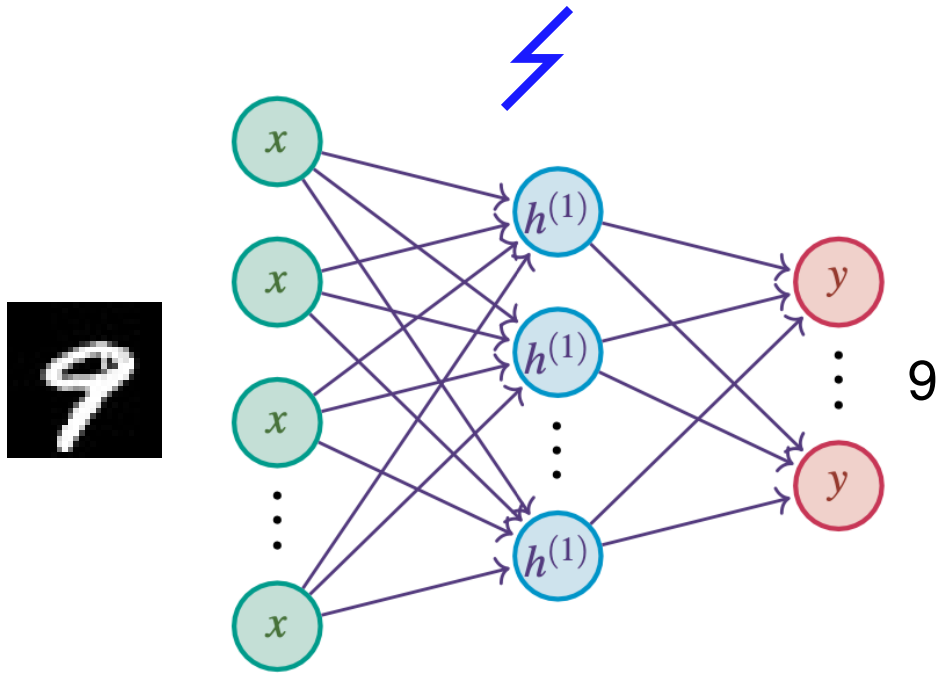
Learning Algorithm

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

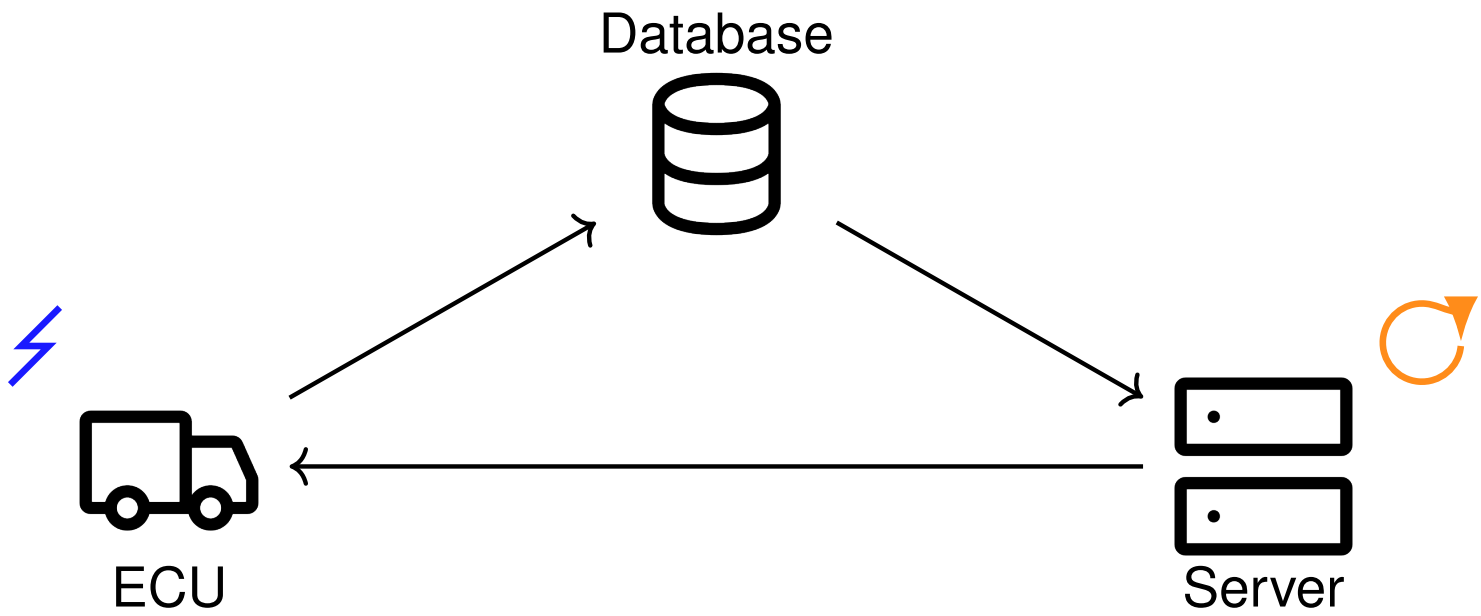
Require: initial weights $w^{(0)}$, number of epochs E, batch size B, training data with T entries
Ensure: final weights $w^{(E*T)}$

```
for e = 0 → E − 1 do
    for b = 0 → T/B do
        for t = b ∗ B → (b + 1) ∗ B do
            estimate ∇ℒ(w^(t))                    ▷ ℒ here is MSE
            compute Δw^(b) += −∇ℒ(w^(t))
        end for
        w^(e+1) := w^(e) + η Δw^(e)
    end for
end for
return w^(T)
```

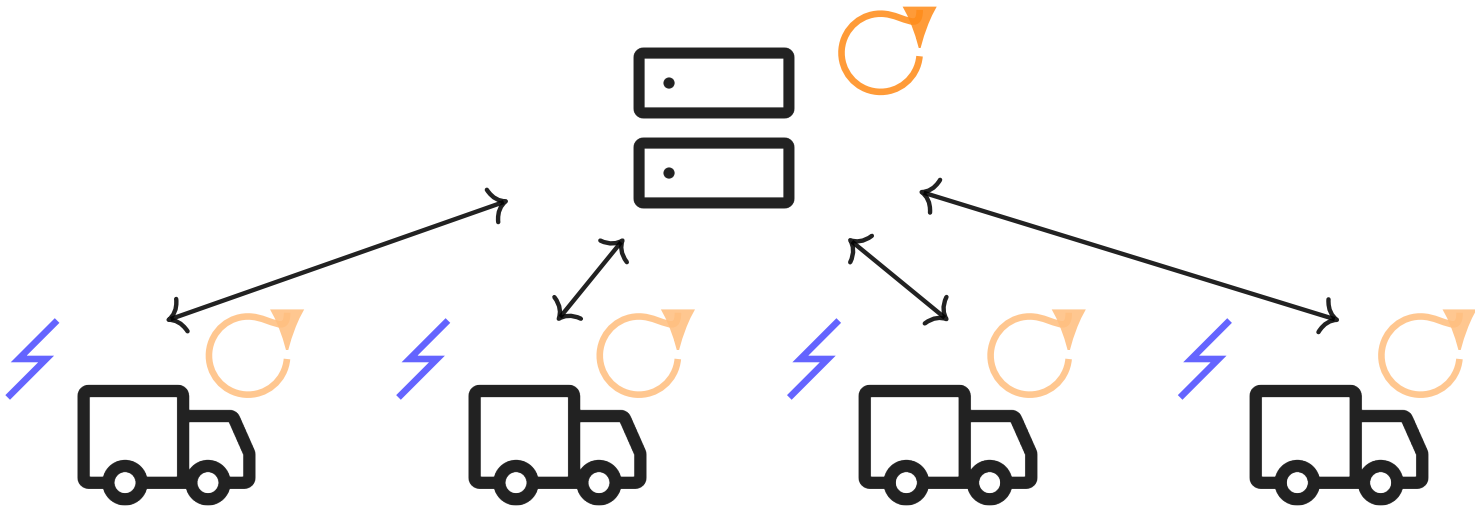
Neural Network Inference



Traditional Paradigm



Federated Learning



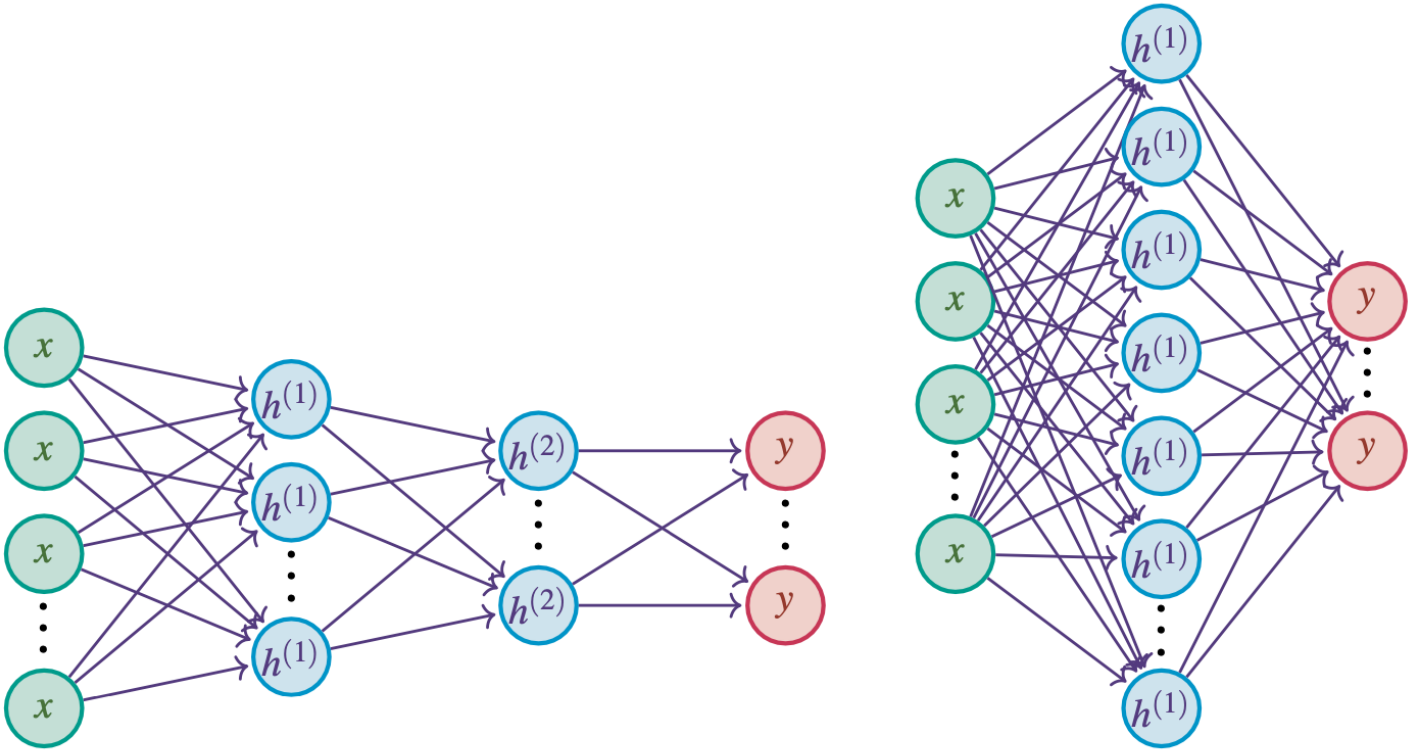
C based HDR-NN

github.com/paperbound/hdrnn-arm

```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;
    }
}
```

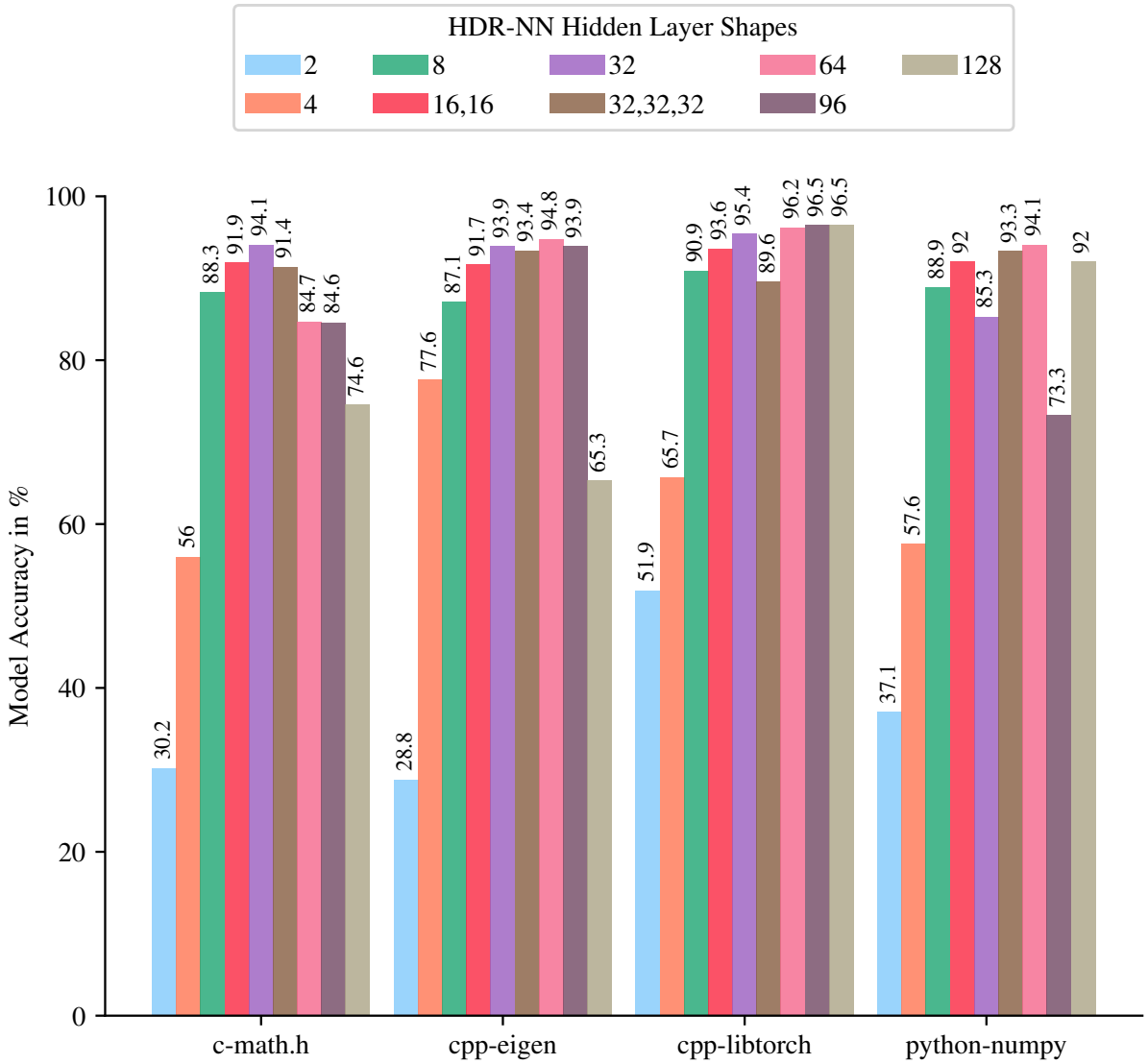
HDR-NN configurability



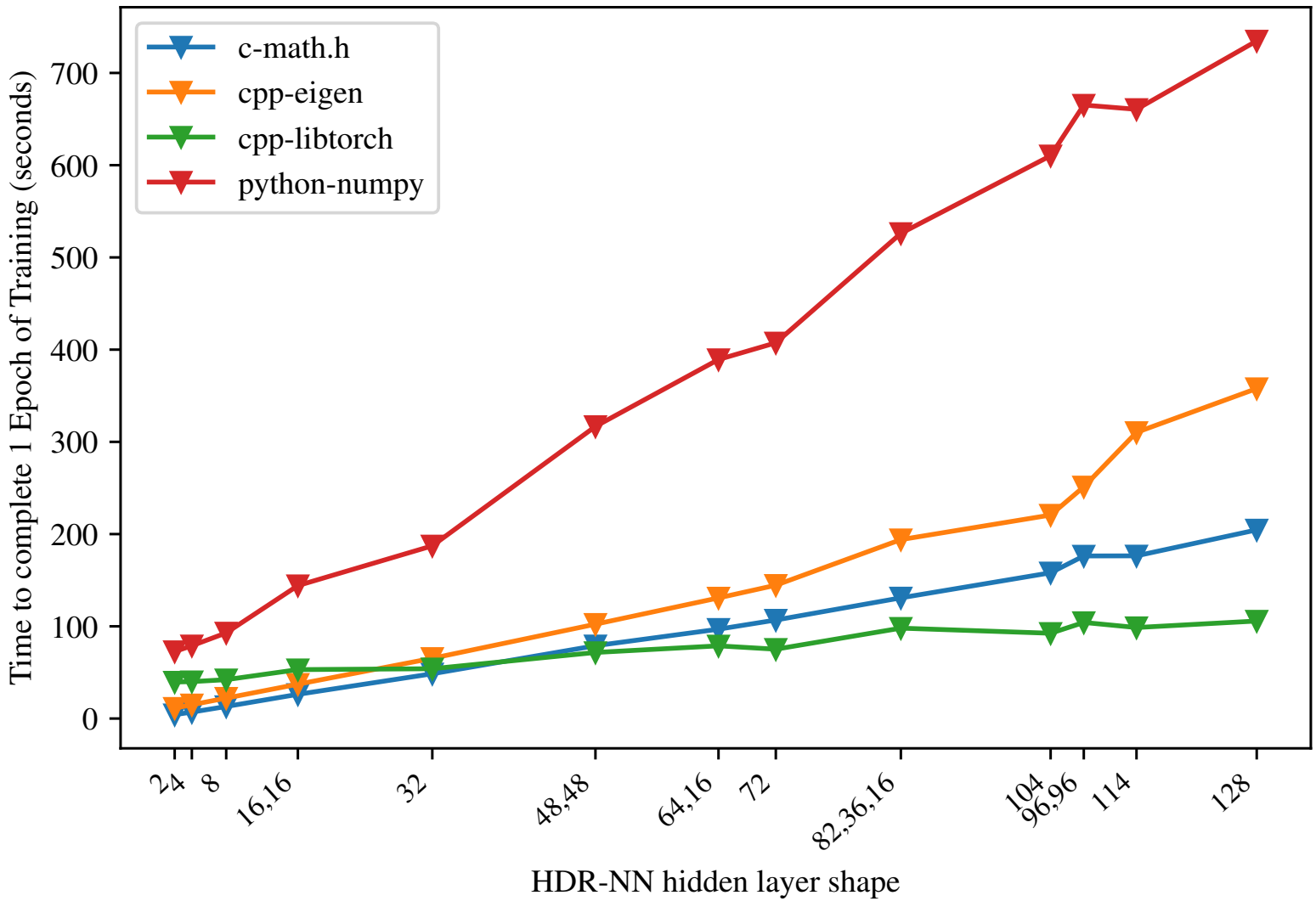
HDR-NN configurability

```
→ ./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]
```

Accuracy


HDR-NN Hidden Layer Shapes
2
4
8
16,16
32
32,32,32
64
96
128
Model Accuracy in %
0
20
40
60
80
100
30.2
56
88.3
91.9
94.1
91.4
84.7
84.6
74.6
28.8
77.6
87.1
91.7
93.9
93.4
94.8
93.9
65.3
51.9
65.7
90.9
93.6
95.4
89.6
96.2
96.5
96.5
37.1
57.6
88.9
92
85.3
93.3
94.1
73.3
92
c-math.h
cpp-eigen
cpp-libtorch
python-numpy

Training Time



Development Journey

- Linux, software packages for MCIMX6Q
- Implementing back propagation in `C`
- Implementing HDR-NN in `C`, `C++ / Eigen`
- Compiling `libtorch` for `armv7hl`

Software Development on Embedded Devices

- Embedded Platforms are fragmented in both *hardware* and *software*
- Challenging to advance software
- Research Approaches : Tiny ML, Tiny Training Engine

Future Work

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