Contents

Part I:
Introduction

Neural network training and inference in embedded environments

Talk about the number of devices in the space of embedded systems / tiny devices (Estimates put the total number of these devices north of 20 billion) and about the potential of running machine learning on that compute. Introduce sustainability notion of utilising those embedded devices for the purpose of performing ML computes as a means of efficient utilisation ... However much of the potential of running machine learning appilations on them remain unattained due to the difficulties in creating these applications

Among the approaches that would be salient on these platforms, neural network approaches are the most sought after owing to the unprecedented progress made in their practical applications. Tiny Machine Learning is a burgeoning field that looks at how this space of embedded devices can be made more suitable to create and explore the potential machine learning applications that it can support. An important feature of machine learning applications are their iterative improvement process. For neural network applications this happens during the training process which traditionally consumes a lot of compute resource

Why perform training and inference on ECUs?

In the world of embedded systems resources such as compute, memory, network bandwith etc. are all limited. The traditional model of sending data from embedded device sensors off-board to compute clusters on the cloud presents several challenges such as bandwith consumption, privacy considerations, and more that makes it attractive to perform both training and inference on-board the embedded device

Federated Learning One approach to making this training loop take place from within these platforms is Federated Learning which cruicially allows for the data to remain on the device

1. Background

Describe ECU systems, Tiny ML

Hypothesis: Training and inference of (small) neural networks in embedded systems can be considerably improved compared to general purpose neural networks frameworks

1.1 Anomally Detection

Explain the problem. Introduce terminlogy that will get explained in the next chapter

1.1.1 Considerations in Embedded Environments

1.2 Scania Embedded Systems

ECU systems, the kind of data they generate, and the potential applications

1.2.1 i.MX6 Target Processor

i.MX6 specifications and constraints

1.3 Development for Embedded Linux

Short introduction to writing applications for embedded linux

1.3.1 Yocto Project

Outline the Yocto Project based Build environment

2. Theory

Theoretical foundations

2.1 Expectations for the Hardware

Layout the Architecture of the i.MX6 - ISA and specifications. Optimisation possibilities through using the SIMD etc

2.1.1 Training vs Inference

Requirements for Training and uploading the wieghts vs Inference

2.2 Neural Network Performance Optimisations Techniques

Contrast traditional implementations in resource rich environments and the constraints of embedded environment. Layout general strategies to acquire performance improvements with little losses to accuracy e.g Purning, Quantisation

2.3 Tiny Machine Learning

Approaches to doing Machine Learning in Embedded Environments

2.4 ARM's CMSIS-NN

Introduction to ARM CMSIS-NN Kernels

Part II: Implementation

Approach to writing Tiny Neural Networks and the nature of their target environments

3. Design

Describe the system design of the Real-Time Linux environment, support provided for the neural network execution, performance engineering based on the hardware, and design and optimisation of the neural network that is executed

NOTE: Talk about the reason why MNIST database was chosen and also elaborate how the idx gunziped version was also avoided to remove the dependency of a gzip library crate / package etc

3.1 Embedded Linux Environment

Information regarding configuration and other details

3.1.1 Neural Network Support

Leveraging hardware support for the application from the OS layer

3.1.2 CMSIS-NN Kernels

Utilising ARM's CMSIS-NN Kernels

3.1.3 Hooks for the Applications

Application design

3.2 Tiny Neural Network

The architecture of the Anomally Detection Neural Network

4. Development

Details of how different neural networks were implemented

4.1 General Distribution of Work

4.2 The C Implementation

4.3 CMSIS-NN Implementation

4.4 Testing on Device

Process for testing on device

4.4.1 Flashing the application

Where the device comes in the development loop

4.4.2 Performance Evaluation

Perf tools and profiling techniques

Part III:
Analysis

Results from the implementation

5. Results

The performance of the different neural networks on the i.MX6 processor

5.1 Neural Network Performance

Performance measure considered. Execution times vs Neural Network Accuracy etc…

5.1.1 ADNN on Tensorflow-Lite

Performance of a CNN application performing Anomally Detection running on tensorflow lite

5.1.2 ADNN written in C

Performance of a similar network written in C

5.1.3 ADNN written using CMSIS-NN

Performance of a similar network utilising CMSIS-NN

5.2 Further Neural Network Optmisations

Further breakdown of the performance achieved from different optimisation techniques

5.2.1 Pruning the Network

6. Discussion

Which optimisation approaches gave the most in improvement?

7. Conclusion and Future Work

What does it all mean? Where do we go from here?

References

[1] Robert David, Jared Duke, Advait Jain, Vijay Janapa Reddi, Nat Jeffries, Jian Li, Nick Kreeger, Ian Nappier, Meghna Natraj, Shlomi Regev, Rocky Rhodes, Tiezhen Wang, and Pete Warden. Tensorflow lite micro: Embedded machine learning on tinyml systems, 2020.